**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

"___" _____202____р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою «Комп'ютерні системи та компоненти»**

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Система виявлення мережевих втручань з використанням машинного навчання

Виконала студентка IV курсу, групи КВ-61
Савосько Олександра Миколаївна _____
(підпис)

Керівник к.т.н.,с.н.с., доцент кафедри СПСКС Боярінова Ю.Є.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М.

(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____

_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та компоненти»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___»_____20___р.

**ЗАВДАННЯ**
**на дипломний проєкт студента**
Савосько Олександра Миколаївна

1. Тема проєкту Система виявлення мережевих втручань з використанням машинного навчання
,
керівник проєкту к.т.н.,с.н.с., доцент Боярінова Юлія Євгенівна,
затверджені наказом по університету від «__»_____ 20__р. №_____

2. Термін подання студентом проєкту

3. Вихідні дані до проєкту Система виявлення мережевих втручань з використанням машинного навчання

4. Зміст пояснювальної записки
   1) Аналіз машинного навчання та існуючих систем виявлення вторгень
   2) Вибір та опис методів машинного навчання
   3) Аналіз набору даних для навчання та тестування моделей та метрик оцінювання їх роботи

4) Створення системи виявлення вторгнень з використанням машинного навчання

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) презентація; структурна схема декомпозиції другого рівня; структурна схема декомпозиції третього рівня; структурна схема принципу беггінгу; схема роботи алгоритму k-NN.

6. Консультанти розділів проєкту

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| нормоконтроль | Ярослав КЛЯТЧЕНКО доц. каф. СПСКС, к.т.н. | | |

7. Дата видачі завдання 4.11.2019

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1. | Видача завдання на дипломне проєктування | 04.11.2019 | |
| 2. | Вивчення літератури за тематикою роботи | 16.11.2019 | |
| 3. | Розроблення та узгодження технічного завдання | 20.11.2019 | |
| 4. | Розроблення структури диплому | 17.01.2020 | |
| 5. | Розроблення моделей | 04.02.2020 | |
| 6. | Навчання моделей | 15.03.2020 | |
| 7. | Тестування моделей | 04.04.2020 | |
| 8. | Підготовка матеріалів текстової частини проєкту | 24.04.2020 | |
| 9. | Підготовка матеріалів графічної частини проєкту | 17.05.2020 | |
| 10 | Оформлення технічної документації прокту | 19.05.2020 | |
| 11 | Попередній розгляд матеріалів на засіданні кафедри | 22.05.2020 | |

Студент _____ Олександра САВОСЬКО
(підпис)

Керівник проєкту _____ Юлія БОЯРІНОВА_____
(підпис)

# АНОТАЦІЯ

Дипломний проєкт складається з 62 сторінок, 44 рисунків, 4 таблиць.

В дипломному проєкту досліджено придатність методів машинного навчання до задачі класифікації мережевих втручань.

У роботі виконано порівняльний аналіз існуючих рішень для виявлення аномалій в мережі. За результатом проведеного аналізу було сформовано функціональні вимоги системи, що розробляється. В дипломному проєкті реалізовано шість оптимальних моделей класифікації з використанням різних алгоритмів машинного навчання,а саме: Рандомний ліс, Дерево рішень, Метод k найближчих сусідів, Метод опорних векторів, Логістична регресія, Градієнтне прискорення. Проведений аналіз створених моделей за допомогою метрик якості: fit_time, score_time, accuracy, f1_weighted, recall, auc_roc.

Дипломний проєкт виконано мовою програмування Python з використанням інструменту для ітерактивної розробки даних у сфері Data Science, а саме Jupyter Notebook, надано можливості використання алгоритмів машинного навчання, для класифікації мережевих втручань, обраний найкращий алгоритм для даної задачі.

Ключові слова: машинне навчання; класифікація; математична модель; логістична регресія; дерево рішень; градієнтне прискорення; рандомний ліс; метод k найближчих сусідів(k-NN); метод опорних вікторів; метрика.

## ABSTRACT

The diploma project consists of 62 pages, 44 images, 4 tables.

The purpose of the diploma project is to explore the applicability of machine learning techniques to the classification of the network interventions.

The comparative analysis of existing analog solutions for identifying network anomalies was conducted in the project. The advantages and disadvantages were described. As results of the analysis functional requirements for the the developed models were formed. Six optimal classification models using different machine learning algorithms such as Random Forest, Decision tree, k Nearest Neighbor, Support Vector Machine,Logistic regression, Gradient Boosting, implemented in the diploma project. Also a quality metrics analysis of the models created by: fit_time, score_time, accuracy, f1_weighted, recall, auc_roc.

As a result of this project, acquired skills in the Python programming language using the iterative tool for development in the field of Data Science, namely Jupyter Notebook. I learned the possibility of using machine learning algorithms to classify network interventions, was selected the best algorithm for this problem.



Key words: machine learning; classification; mathematical model; logistic regression; decision tree; gradient boosting; random forest; k nearest neighbor; support vector machine; metrics.

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.045420.002 ТЗ | Система визначення | 4 | | |
| | | | мережевих втручань | | | |
| | | | з використанням | | | |
| | | | машинного навчання | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045420.004 ПЗ | Система визначення | 62 | | |
| | | | мережевих втручань | | | |
| | | | з використанням | | | |
| | | | машинного навчання | | | |
| | | | | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045420.006 Д1 | Схема декомпозиції | 1 | | |
| | | | третього рівня | | | |
| | | | Схема структурна; | | | |
| | | | | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045420.007 Д1 | Схема принципу | 1 | | |
| | | | беггінгу | | | |
| | | | Схема структурна; | | | |
| | | | | | | |
| | | | | | | |

| Змін. | Арк. | № докум. | Підпис | Дата | ІАЛЦ.045420.001 ОА | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Савоcько О.М. | | | Система визначення мережевих втручань з використанням машинного навчання<br>**Опис альбому** | Літ. | Аркуш 1 / Аркушів 2 |
| Перевірив | | Боярінова Ю.Є. | | | | | |
| Консульт. | | | | | | КПІ ім. Ігоря Сікорського, ФПМ КВ-61 | |
| Н. контроль | | Клятченко Я.М. | | | | | |
| Зав. каф. | | Романкевич В.О. | | | | | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.045420.005 Д1 | Схема роботи | 1 | | |
| | | | Алгоритму k-NN | | | |
| | | | Схема алгоритму | | | |
| | | | | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045420.005 Д1 | Схема декомпозиції | 1 | | |
| | | | другого рівня | | | |
| | | | Схема структурна; | | | |

| Змін. | Арк. | № докум. | Підпис | Дата | ІАЛЦ.045420.001 ОА | Арк. 2 |
|---|---|---|---|---|---|---|

# ЗМІСТ

## 1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Система визначення мережевих втручань з використанням машинного навчання».

Галузь застосування: створення системи визначення мережевих втручань з використанням машинного навчання.

## 2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

## 3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є покращення способів виявлення мережевих втручань з використання методики класифікації та оцінювання якості роботи моделей.

## 4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет.

# 5. ТЕХНІЧНІ ВИМОГИ

## 1.1 Вимоги до програмного продукту, що розробляється

- стійкість системи виявлення вторгнень до існуючих та невідомих мережевих атак;
- незалежність результатів класифікації від конкретного набору даних моделі;
- швидка та якісна робота алгоритмів;
- наявність декількох метрик оцінювання моделей;
- структура моделі повинна бути блочною, тобто допускати можливість заміни, додавання і виключення деяких частин без переробки всієї моделі;
- тривалість навчання моделі повинна бути по можливості мінімальною.
- відображення аналітичної та графічної інформації про дану модель класифікації.

## 5.2 Вимоги до апаратного забезпечення

- процесор з тактовою частотою від 2 ГГц ;
- вільний дисковий простір не менш ніж 128 Гб.

## 5.3 Вимоги до програмного та апаратного забезпечення користувача

- Інтерфейс повинен коректно працювати у сучасних веб-браузерах Opera та Google Chrome;
- наявність доступу до мережі Wi-Fi (IEEE 802.11 b/g/n).

## 6. ЕТАПИ РОЗРОБКИ

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів |
|---|---|---|
| 1. | Видача завдання на дипломне проєктування | 04.11.2019 |
| 2. | Вивчення літератури за тематикою роботи | 16.11.2019 |
| 3. | Розроблення та узгодження технічного завдання | 20.11.2019 |
| 4. | Розроблення структури диплому | 17.01.2020 |
| 5. | Розроблення моделей | 04.02.2020 |
| 6. | Навчання моделей | 15.03.2020 |
| 7. | Тестування моделей | 04.04.2020 |
| 8. | Підготовка матеріалів текстової частини проєкту | 24.04.2020 |
| 9. | Підготовка матеріалів графічної частини проєкту | 17.05.2020 |
| 10. | Оформлення технічної документації проєкту | 19.05.2020 |
| 11. | Попередній розгляд матеріалів на засіданні кафедри | 22.05.2020 |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.045420.004 ПЗ | Система визначення | 62 | | |
| | | | втручань з використанням | | | |
| | | | машинного навчання | | | |
| | | | | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045420.006 Д1 | Схема декомпозиції | 1 | | |
| | | | третього рівня | | | |
| | | | Схема структурна; | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045420.007 Д1 | Схема принципу | 1 | | |
| | | | беггінгу | | | |
| | | | Схема структурна; | | | |
| | | | | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045420.005 Д1 | Схема роботи | 1 | | |
| | | | Алгоритму k-NN | | | |
| | | | Схема алгоритму | | | |
| | | | | | | |

| Змін | Арк. | № докум. | Підпис | Дата | ІАЛЦ.045420.003 ТП | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Савоcько О.М. | | | Система визначення мережевих втручань з використанням машинного навчання<br>**Відомість технічного проєкту** | Літ. | Аркуш 1 / Аркушів 2 |
| Перевірив | | Боярінова Ю.Є. | | | | | |
| Консульт. | | | | | | КПІ ім. Ігоря Сікорського, ФПМ КВ-61 | |
| Н. контроль | | Клятченко Я.М. | | | | | |
| Зав. каф. | | Романкевич В.О. | | | | | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.045420.005 Д1 | Схема декомпозиції | 1 | | |
| | | | другого рівня | | | |
| | | | Схема структурна; | | | |

| Змін. | Арк. | № докум. | Підпис | Дата | ДП. 045420.003 ТП | Арк. 2 |
|---|---|---|---|---|---|---|

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Градієнтне прискорення - Gradient Boosting(GB)

Дерева рішень - Decision tree(DT)

Логістична регресія - Logistic Regression(LR)

МН- машинне навчання

Метод опорних вікторів - Support Vector Machine(SVM)

Метод k найближчих сусідів - k Nearest Neighbor(k-NN)

Рандомний ліс – Random Forest(RF)

СВВ- системи виявлення вторгнень

API- application programming interface

AUC – Area under ROC Curve або площа під кривою помилок;

FPR- false positive rate

KDD – Knowledge Discovery and Data Mining

NSL-KDD-Network Services Library

ROC – receiver operating characteristic або крива помилок

SVC- Support Vector Classifier

TPR- true positive rate

# ВСТУП

Інтенсивний розвиток Інтернет, повсюдний перехід на електронні форми зберігання і передачі інформації, активне впровадження в повсякденне життя електронних форм платежів і багато інших чинників сьогоднішньої реальності вплинули на те, що безпека мереж і мережевих сервісів стала дійсно нагальною проблемою. Важлива інформація може бути втрачена чи викрадена зловмисниками.

Продумана і добре організована система виявлення мережевих втручань дозволить уникнути або звести до мінімуму втрату важливих даних.

У будь-який момент часу в мережі одночасно може бути багато клієнтів, і вони, природно, виробляють великий трафік. Кожне мережеве з'єднання можна візуалізувати як набір атрибутів. Дані про трафік можна реєструвати та використовувати для вивчення та класифікації на нормальний та ненормальний. Щоб обробляти об'ємну базу даних, найкраще буде використати методики МН(машинне навчання).

Інтелектуальні системи виявлення вторгнень можуть бути побудовані лише за наявності ефективного набору даних. Набір даних (Data set) із значною кількістю якісних даних, може допомогти навчитися перевіряти вторгнення системі виявлення. Набір даних NSL-KDD(Network Services Library Knowledge Discovery and Data Mining) - це вдосконалена версія попереднього набору даних KDD ‘‘99(Knowledge Discovery and Data Mining). У цій роботі набір даних NSL-KDD аналізується та використовується для вивчення ефективності різних алгоритмів класифікації при виявленні мережевих втручань.

Класифікація - одна з найпопулярніших методик МН для прогнозування класу нових зразків, використовуючи модель, що випливає з навчальних даних. Взагалі класифікація визначається як метод навчання, який відображає або класифікує екземпляри даних у відповідні мітки класу, які окреслені у визначеному наборі даних. Існує багато алгоритмів класифікації, розроблені, щоб перевершити один одного.

В даній роботі досліджено способи придатності методів лінійної класифікації до задачі класифікації мережевих втручань. Вивчено шість популярних методик класифікації машинного навчання: LR (логістична регресія), k-NN (метод k найближчих сусідів), DT (дерево рішень), RF (рандомний ліс), SVM (метод опорних векторів), GB (градієнтне прискорення), досліджено простір параметрів і оцінено їх роботу за допомогою математичних метрик.

# 1. АНАЛІЗ МАШИННОГО НАВЧАННЯ ТА ІСНУЮЧИХ СИСТЕМ ВИЯВЛЕННЯ ВТОРГЕНЬ

## 1.1. Основні поняття машинного навчання

Неформально МН (машинне навчання) можна уявити як процес знаходження невідомого вирішального правила (або невідомої цільової функції) за допомогою деякої початкової інформації, яка не є повною. Цю неповну початкову інформацію називають навчальною.

Кажуть, що значення аргументів шуканої функції в даній точці в сукупності є описом точки (об'єкта) в деякій проблемній області. Якщо такий опис є правильним за змістом розв'язуваної задачі, то таку точку разом з її описом називають допустимим об'єктом. У завданнях МН аргументи цільових функцій (вирішальних правил) і, відповідно, їх допустимі описи можуть бути різноманітними. На відміну від класичних математичних задач, можуть використовуватися допустимі описи зображень, текстів, структур даних і багато іншого. Це призводить до того, що при вирішенні задач МН використовуються різні розділи математики, які підходять в конкретних випадках.

Навчальна інформація, як правило, являє собою кінцеву сукупність прикладів – допустимих точок (описів) разом зі значеннями цільової функції в цих точках. В цьому випадку навчальну інформацію називають навчальною вибіркою. Якщо МН припускає знаходження кінцевого набору невідомих характеристичних функцій множин, то таке навчання зазвичай називають навчанням розпізнаванню. Якщо цільова функція приймає тільки

два значення, то її називають класифікуючою або класифікатором. Якщо цільова функція приймає довільні значення, то її називають регресією.

Розглянемо головним чином клас задач алгоритмічного навчання класифікації – самий загальний випадок МН. Це передбачає побудову та використання алгоритмів навчання і алгоритмів класифікації, отриманих в результаті навчання.

Будемо говорити, що узгодженої з навчальною інформацією або коректною називається будь–яка функція, яка в точках, що входять в цю навчальну інформацію, приймає точно такі ж значення, які містяться в прикладах з цієї навчальної інформації.

Алгоритм навчання називається коректним на навчальній вибірці, якщо він видає узгоджену із заданою навчальною інформацією обчислювану функцію.

## 1.2. Аналіз існуючих рішень

Метою цього огляду є оцінка того, що відомо з попередніх досліджень мережевих втручань, а також пропонування перспективних нових напрямків для майбутніх досліджень та застосувань.

Мережеві атаки різноманітні за своєю структурою і складністю виявлення. Завдання протидії атакам є важливою для коректного функціонування систем і запобігання порушенню безпеки. Виявити атаку можна двома методами [1]:

1) Сигнатурний метод зводиться до пошуку ознак вже відомих атак. Даний метод базується на порівняння поточного стану системи з сигнатурою. Сигнатура – це купа умов, при яких настає подія, тобто атака або вторгнення [2]. Перевага сигнатурного методу в тому, що він практично

не схильний до помилкових спрацьовувань. Недоліком цього методу є неможливість виявляти незакладені в систему атак.

2) Метод пошуку аномалій дозволяє реагувати на раніше невідомі атаки, але схильний до помилкових спрацьовувань і вимагає точного налаштування для кожного об'єкта, що спостерігається. Базується на визначенні умов нормальної поведінки системи, при їх невиконанні поведінка буде вважатися аномальною. Аномальна поведінка зазвичай є наслідком впливу зловмисників.

Загальні схеми роботи вищезгаданих методів зображені нижче (рисунок1.1, рисунок1.2 ):

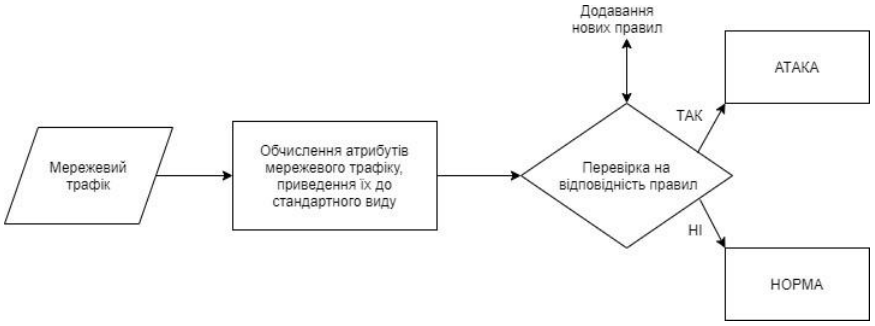


Рисунок 1.1 – Схема роботи методів виявлення зловживань

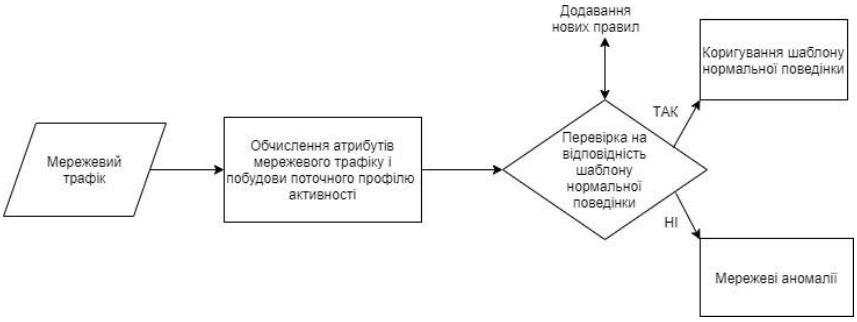


Рисунок 1.2 – Схема виявлення аномалій

Кожний з цих методів має свої переваги і недоліки, тому в більшості існуючих СВВ (системи виявлення вторгнень) застосовуються комбіновані рішення, засновані на синтезі відповідних методів.

Щоб спростити процес усунення аномалій, їх слід систематизувати. Через те що існує величезна кількість різновидів аномалій,відсутній стандартний підхід щодо їх класифікації. В статті [4] представлено варіант класифікації аномалій, як об'єкту впливу.

Методи виявлення аномалій, як і сигнатурні методи, включають в себе поведінкові методи, методи МН і методи штучного інтелекту [3] (рисунок1.3). Поведінкові методи ґрунтуються на порівнянні поведінки поточного об’єкта системи з моделлю нормальної поведінки.

Ще можна виділити дві групи методів: з контрольованим навчанням, і з неконтрольованим навчанням («навчання без вчителя»). Основна відмінність між ними полягає в тому, що методи контрольованого навчання використовують фіксований набір параметрів оцінки і апріорні відомості про значення параметрів оцінки. Час навчання фіксований. У неконтрольованому ж навчанні безліч параметрів оцінки може змінюватися з плином часу, а процес навчання відбувається постійно.

Сучасні СВВ(системи виявлення вторгнень) можуть виявляти несанкціоновані дії в мережі і окремих її вузлах та машинально реагувати на них практично, все це в реальному масштабі часу. СВВ ще й аналізують поточні події, порівнюють їх з тими, що вже здійснилися, для того,щоб ідентифікувати атаки і прогнозувати майбутні події [5].

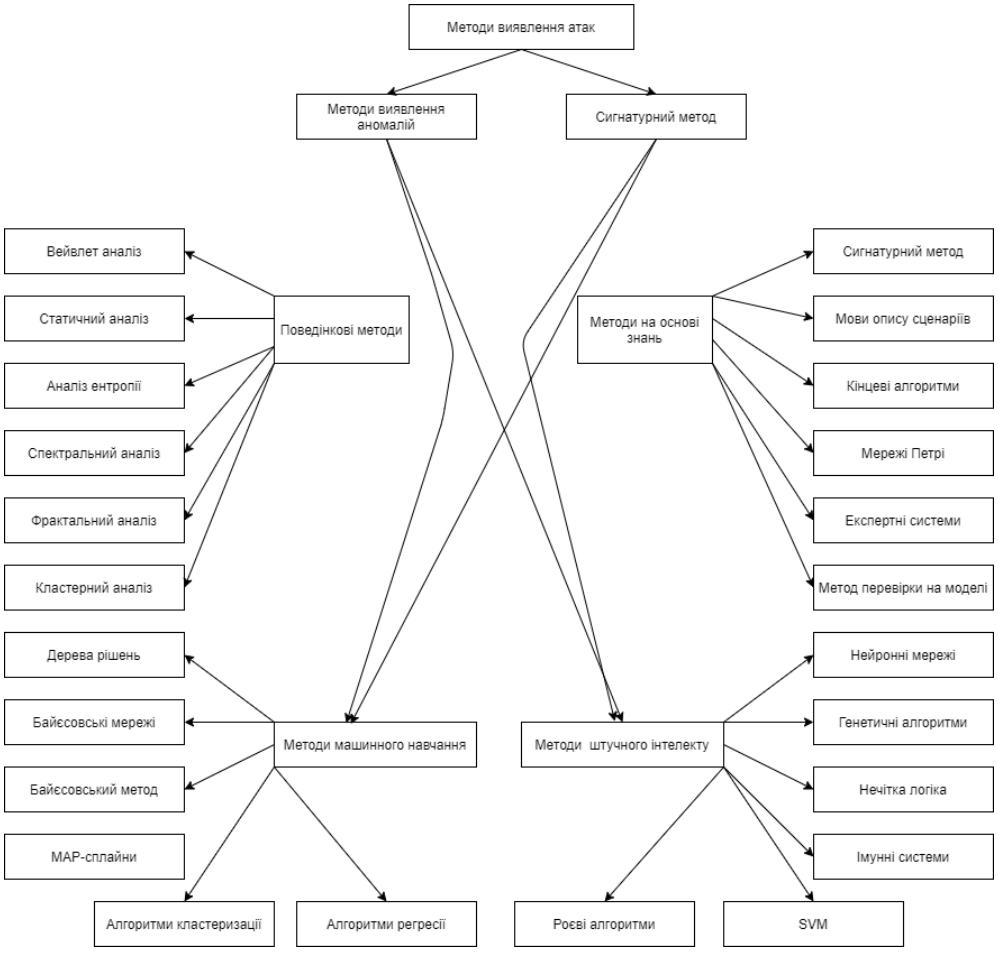


Рисунок 1.3- Загальна схема методів виявлення мережевих атак [5]

Реалізовані в даний час в СВВ методи засновані на загальних уявленнях теорії розпізнавання образів. Для побудови СВВ існує два основних підхода. Перший заключається в аналізі пакетів, що передаються мережею, а другий в аналізі журналів реєстрації операційних систем. На сьогоднішній день проводяться спроби в розробці перспективної, але дуже складної в реалізації й застосуванні експертної системи, яка буде аналізувати мережевий трафік, навчатися по ньому і завдяки цьому виявляти аномалії. Для виявлення аномалії на основі експертної оцінки формується образ нормального функціонування інформаційної системи.

Цей образ виступає як сукупність значень параметрів оцінки. Його зміна вважається проявом аномального функціонування системи. Після виявлення аномалії і оцінки її ступеня формується судження про природу змін: чи є вони наслідком вторгнення або допустимим відхиленням. Для виявлення зловживань також використовується образ (сигнатура), однак тут він відображає заздалегідь відомі дії атакуючого. Основна проблема в використанні експертних систем полягає в великій кількості помилкових спрацьовувань. Для вирішення цієї проблеми необхідне точне попереднє налаштування[6].

Використання СВВ має ряд своєрідних особливостей. Це пов’язано з відменними типами систем і шляхами їх застосуваннь.

Недоліки сучасних СВВ можна розділити на дві групи - недоліки, пов'язані зі структурою СВВ, і недоліки, що відносяться до реалізованих методів виявлення.

Недоліки структур СВВ:

1. Відсутність загальної методології побудови. Частково це можна пояснити недостатністю спільних угод в термінології, так як СВВ - це досить новий напрямок [7].
2. Ефективність. Часто методи системи намагаються виявити будь-яку зрозумілу атаку, що призводить до ряду незадовільних наслідків.
3. Портативність. До цих пір більшість СВВ створюється для використання на конкретному обладнанні, і досить важко використовувати їх в іншій системі, де потрібно реалізувати схожу політику безпеки. Основною причиною цього є те, що багато СВВ спостерігають за певними пристроями, програмами конкретної ОС. Також слід зауважити, що кожна ОС розробляється для виконання

конкретних завдань. Отже, переорієнтувати СВВ на інші ОС досить складно, за винятком тих випадків, коли ОС розроблені в якомусь загальному стилі.

4. Право на оновлення. Дуже складно відновити існуючі системи новими технологіями виявлення. Нова підсистема повинна взаємодіяти з усією системою, і часом неможливо забезпечити універсальну можливість взаємодії.
5. Для установки СВВ дуже часто потрібні додаткові навички, істотно відмінні від навичок в області безпеки.
6. Продуктивність і допоміжні тести - важко оцінити продуктивність СВВ в реальних умовах. Більш того, відсутній загальний набір правил для тестування СВВ, на підставі яких можна було сказати про доцільність використання даної системи в конкретних умовах і отримати якісь кількісні показники.

   Недоліки методів виявлення:

   - неприпустимо високий рівень помилкових спрацьовувань і пропусків атак;
   - слабкі можливості по виявленню нових атак;
   - більшість вторгнень неможливо визначити на початкових етапах;
   - важко, іноді неможливо, визначити атакуючого, цілі атаки;
   - відсутність оцінок точності і адекватності результатів роботи;
   - неможливо визначати «старі» атаки, що використовують нові стратегії;
   - складність виявлення вторгнень в реальному часі з необхідною повнотою в високошвидкісних мережах;

- слабкі можливості з автоматичного виявлення складних координованих атак;
- значне перевантаження систем, в яких функціонують СВВ, при роботі в реальному часі.

На основі викладеного можна зробити висновок про те, що в практичній діяльності накопичений значний досвід вирішення проблем виявлення вторгнень. Застовувані СВВ в значній мірі засновані на емпіричних схемах процесу виявлення вторгнень, подальше вдосконалення СВВ пов'язано з конкретизацією методів синтезу та аналізу складних систем, теорії розпізнавання образів в застосуванні до СВВ.

### 1.3. Вибір базових класифікаторів

Вирізняють контрольоване навчання (з вчителем), неконтрольоване навчання (без вчителя), напівконтрольоване (напівавтоматичне) та навчання з підкріпленням (рисунок 1.4) .

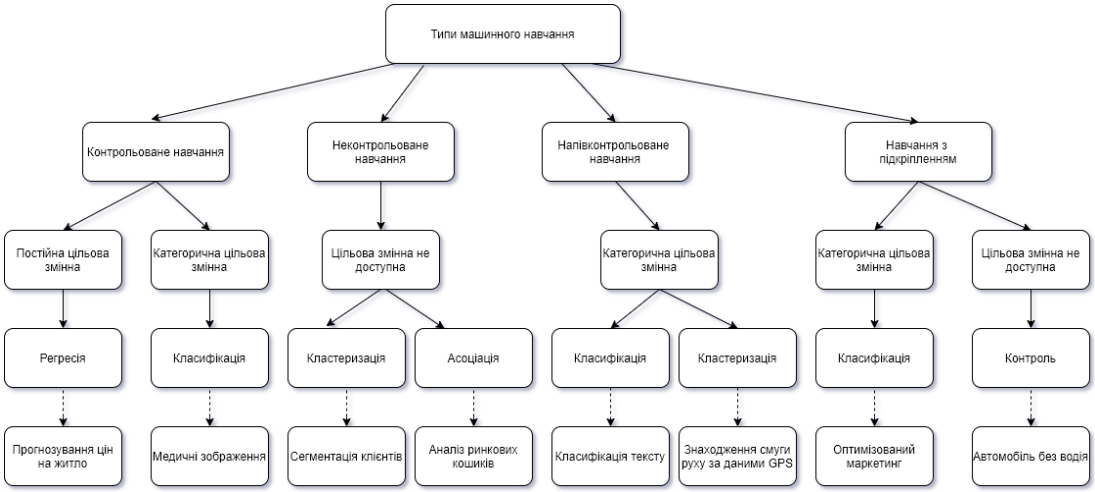


Рисунок 1.4 – Типи задач МН

Класифікація - це одна з найпопулярніших методик МН для передбачання класу нових зразків, використовуючи модель, що витікає з навчальних даних. В цілому класифікація формулюється так: на метод навчання, який відтворює або упорядковує екземпляри даних у належні мітки класу, які визначені у визначеному наборі даних.

Відповідно до [8], задача навчання по прецедентам формально має такий вигляд:

Нехай X – множина об'єктів, Y – множина відповідей, X → Y – невідома залежність (target function). Дано:

$\{x_1, \dots, x_l\} \subset X -$ навчальна вибірка (training sample);

$y_i = y(x_i), i = 1, \dots, l$ –це відповіді, що відомі.

Знайти:

: X функцію (алгоритм) прийняття рішення (decision function), що приближує на всій множині X.

Розглянемо детальніше, що можуть собою представляти об'єкти та відповіді.

$f_j: X \to D_j, j = 1, \dots, n$ –ознаки об'єктів (features).

Типи ознак:

- бінарна ознака $f_j - D_j = \{0,1\}$
- номінальна ознака $f_j - |D_j| < \infty$;
- порядкова ознака $f_j - |D_j| < \infty, D_j$ впорядкована
- кількісна ознака $f_j - D_j = R$;

Вектор $(f_1(x), \dots, f_n(x))$ називають вектором ознак об'єкта x.

Розглядають матрицю ознак об’єктів (feature data)

$$F = \begin{pmatrix} f_1(x_1) \dots f_n(x_1) \\ f_1(x_l) \dots f_n(x_l) \end{pmatrix}$$

Для задач класифікації (classification):

- $Y = \{-1; +1\}$ –класифікація на 2 класи.
- $Y = \{1, \dots, M\}$ – на М класів, що не можуть перетинатись.
- $Y = \{1,0\}^M$ – на М класів, що можуть перетинаютьсь.

Класифікація даних - це двоетапний процес:

1) крок навчання, де з заданого набору даних будується модель класифікації, а дані, з яких вчиться функція або модель класифікації, називаються навчальним набором;

2) етап класифікації, в якому модель застсовується для тестування або прогнозування міток класу для нових даних, а набір даних, що використовується за для перевірки спроможності до класифікації навченої моделі або функції, називається набором тестування.

Алгоритми класифікації вільно застосовуються в усіх сферах життя людини. Найбільш розповсюдженими є застосування в медицині, розпізнаванні образів, мови та символів, маркетингу, скорингових моделях (оцінки кредитоспроможності), для пошуку спаму та навіть в геології.

# 2. ВИБІР ТА ОПИС МЕТОДІВ МАШИННОГО НАВЧАННЯ

## 2.1. Логістична регресія

LR отримала свою назву вона через функцію, що лежить в її основі, а саме логістичної або сигмоїдної функції (рисунок 2.1), яка була розроблена статистами для опису росту популяції в екології. Вона може приймати будь-яке значення між 0 та але ніколи не дорівнює їм 1 і має форму літери S.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

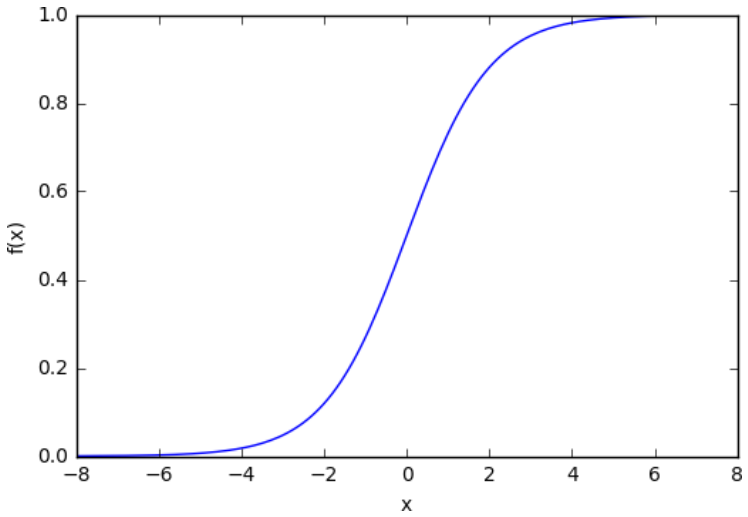


Рисунок 2.1 – Логістична або сігмоїдна функція

Попри назву "Регресія", це не регресія, а класифікаційна модель. Він використовує логістичну функцію для кадрування бінарної вихідної моделі.

LR є подібною з лінійною регресією, але в цьому методі не проводиться пророкування значення числової змінної виходячи з вибірки вихідних значень. Замість цього, значенням функції є ймовірність того, що дане початкове значення належить до певного класу (рисунок2.2).

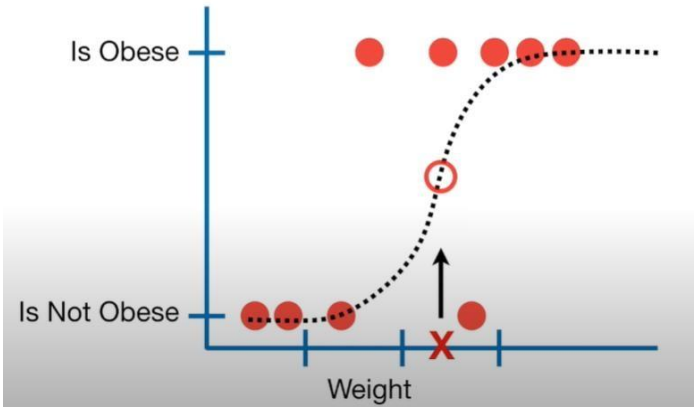


Рисунок 2.2- Ймовірність приналежності до класу

LR може працювати з безперервними даними (як вага та вік) та дискретними даними (як генотип та астрологічний знак). Ми також можемо перевірити, чи корисна кожна змінна для прогнозування (напр. ожиріння). Однак, ми не можемо легко порівняти складну модель з простою моделлю.

Натомість можна просто перевірити, чи впливає змінна на передбачення значно більше від 0. Якщо ні, це означає, що змінна не допомагає передбаченню. Знак Зодіаку - це "марність".Це означає, що ми можемо заощадити час та простір у нашому дослідженні, залишивши цю змінну поза межами.

Здатність LR забезпечувати ймовірності та класифікувати нові зразки за допомогою безперервних та дискретних вимірювань робить її популярним методом МН.

## 2.2. Метод опорних вікторів

SVM один з найпоширеніших методів навчання з вчителем, який представляє екземпляри як набір точок двох типів у N розмірному місці та генерує (N - 1) розмірний гіперплан для поділу цих точок на 2 групи.

Навчання SVM відбувається за допомогою алгоритму послідовної мінімальної оптимізації. Відповідає кусково-лінійній апроксимації:

$$[M < 0] \leq (1 - M)_+$$

Основна ідея роботи SVM:

- на початку дані у реально низькому вимірі;
- далі перенесимо дані у простори більш високої розмірності (регуляризація);
- і знаходимо SVC, який розділяє дані більш високих розмірностей на дві групи (навчання);

Для SVM використовуємо регуляризації із квадратичною нормою. Це є обов'язковою умовою.Навчання виконується на основі спеціалізованих методів квадратичного програмування.

Існують ще й нелінійні модифікації SVM. Їх лінійність чи нелінійність визначається типом ядра.

## 2.3. Метод k найближчих сусідів(k-NN)

k-NN - непараметричний метод, одна з найпростіших методик використання МН.

Метричний алгоритм класифікації відшукуємо в наступному вигляді:

$$a(x; X^l) = \arg\max \sum_{i=1}^{l} [y^{(i)} = y]\, w(i, x)$$

$w(i,x) -$ це ваговий коефіцієнт, він вказує на степінь важливості і-того сусіда об'єкта х,завжди є невід'ємним та по і не зростає.

$$\sum_{i=1}^{l} [y^{(i)} = y]\, w(i, x) -$$

це оцінка близькості об'єкта х до класу у.

Якщо $w(i,x) = [i \leq 1]$, то маємо метод найближчого сусіда.

Основна логіка k-NN полягає в дослідженні сусідніх точок(сусідів), об'єкт відноситься до класу, якому належить більшість його найближчих точок (сусідів).

k-NN в основному включає два гіперпараметри: значення k та функцію відстані.

Значення k: кількість сусідів для участі в алгоритмі k -NN. k має бути налаштовано на основі помилки перевірки.

Функція відстані: зазвичай використовується евклідова відстань, як функція подібності. Як варіант, можуть використовуватись Манхеттенська відстань, відстань Хеммінг, відстань Міньковського.

На наведеній діаграмі (рисунок 2.3) жовті та фіолетові точки відповідають навчальним даним класів А та В. Червона зірка вказує на тестові дані, які підлягають класифікації. коли k = 3, ми прогнозуємо клас В як вихід, а коли K = 6, ми прогнозуємо клас А як вихід.

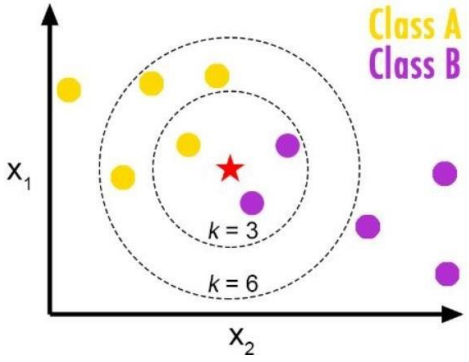


Рисунок 2.3 – Розроділення навчальних даних на діаграмі

Навчання в k-NN немає. Під час тестування k сусідів з мінімальною дистанцією візьмуть участь у класифікації.

## 2.4. Дерева рішень

DT - алгоритм на основі побудови дерева, який використовується для вирішення проблем регресії та класифікації. Як зазначає в своїх лекціях проф Воронцов [9], деревом називають скінченний зв'язний граф з множиною вершин V, що не містить циклів і має виділену вершину $v_0 \in V$,

в яку не входить не одне ребро (корінь дерева). DT походить від незалежних змінних, причому кожен вузол має умову над ознакою. Вузли вирішують, до якого вузла рухатися далі, виходячи з умови. Після досягнення вузла листів прогнозується вихід.

На наведеній діаграмі (рисунок 2.4) ми можемо побачити дерево рішень, що базується на даних про ранжирування, де 1 дуже голодний, а 2 - помірно голодний. Класифікація може бути категоріями або числовими значеннями.Це прийнятно, якщо з різних боків різний порядок питань. Остаточні класифікації можна повторювати.

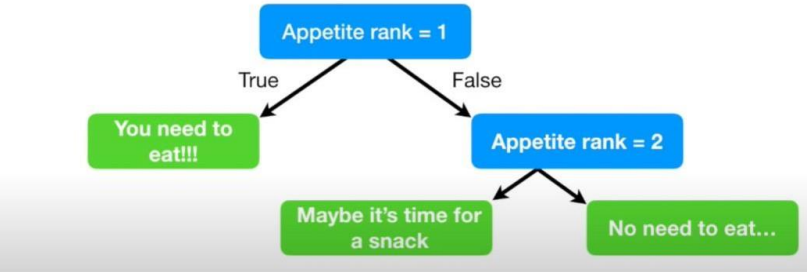


Рисунок 2.4- Дерево з набором внутрішніх вузлів

Використовується поняття ентропії або кількості інформації за Шенноном:

$$S = -\sum_{i=1}^{N} p_i log_2 p_i,$$

де ймовірність знаходження системи в стані i.

Серед інших критеріїв якості розбиття в задачах класифікації виділяють:

Невизначенність Джині (Gini impurity):

$$G = 1 - \sum_k (p_k)^2.$$

Атрибут з максимальним значенням джині вибирається як наступна умова на кожній фазі створення дерева рішень. Якщо значення нерівномірно змішане, бал джині буде максимальним.

Ентропія та невизначеність Джині дають приблизно однакові результати роботи.

## 2.5. Рандомний ліс

МН - це дослідницький процес, де не існує рішення, яке відповідає всім моделям. Зокрема, не існує моделі, яка б мала досягти найвищої точності для всіх доменів, типів проблем або наборів даних [10]. Модель з найкращими показниками варіюється від однієї проблеми до іншої на основі характеристик змінних та спостереженні. Це призводять до виникнення ідеї щодо гібридизованого підходу із залученням деякої техніки оптимізації. Гібридизація повинна враховувати лише важливі особливості існуючого алгоритму що може добре працювати в проблемній області іншого алгоритму та з доступним набором даних.

Більшість прийомів в прикладному ансамблірованні направлено на те, щоб ансамбль був досить різноманітним, тоді помилки окремих алгоритмів на окремих об'єктах будуть компенсуватися коректною роботою інших алгоритмів. По суті, при побудові ансамблю підвищують якість та різноманітність (diversity) базових алгоритмів.

Побудувати ансамбль моделей можна з використанням декількох методів, але головні - це бустінг і беггінг.

Бегінг лежить в основі одного з класичних алгоритмів класифікації., а саме RF. Метод RF являє собою поліпшення методу Дерева рішень з використанням беггінга, яке полягає в усуненні кореляції між деревами.

Всі дерева будуються незалежно (рисунок 2.5) за наступною схемою: вибирається підвибірка навчальної вибірки для поточного дерева шляхом відбору зразків (samples) із заміною. Так близько однієї третини випадків залишаються поза зразком, їх називають “out-of-bag”. Ці дані використовуються для отримання неупередженої оцінки

відхилення по мірі того, як нові дерева додаються до лісу. Кожен зразок, що не був використаний при побудові чергового дерева, пропускається через алгоритм, щоб отримати класифікацію. Так, класифікація об’єктів тестової вибірки проводиться в середньому третиною дерев. В кінці пробігу для кожного об’єкта обирається той клас, який отримав більшу кількість голосів кожного разу, коли цього зразка не було в навчальній вибірці. Частка випадків, коли дерево класифікувало неправильно, усереднена по загальній кількості випадків і це дає нам оцінку похибки. Цей підхід також використовується для отримання оцінок важливості змінних.

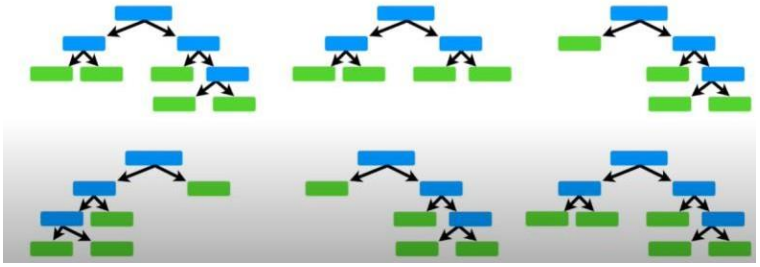

Рисунок 2.5- Приклад RF

Таких дерев можна побудувати дуже велику кількість, кожен раз випадковим чином обираючи новий кореневий вузол, а також нові варіанти комбінації підмножин, які берем з різних стовпців.Дерево будується, як правило, до вичерпання вибірки.

Зміни на кожному кроці призводять до найрізноманітніших дерев. Різноманітність - це те, що робить випадкові ліси ефективнішими, ніж окремі дерева рішень. RF пропонує надійну та точну модель, яка може

обробляти великі різновиди вхідних даних із двійковими, категоричними та безперервними функціями.

Використовується ентропію та оцінка Джині, щоб обчислити значення втрат наборів даних. Я пояснила це в попередньому розділі про дерева рішень ( див. 2.4)

## 2.6. Градієнтне прискорення

Як і RF, GB - це ще одна методика виконання завдань з керованого МН, як класифікація та регресія, з використанням ансамблювання.

Ця технологія, створює модель прогнозу у вигляді множини слабких моделей, представлених, зазвичай, у вигляді DT[11]. Загальна ідея полягає в тому, що випадки, які важко передбачити правильно («важкі» випадки), будуть зосереджені на навчанні, щоб модель вчилася на минулих помилках. Оновлення прогнозів відбувається так, щоб прогнозовані значення були близькі до фактичних, а сума залишків була мінімальною (рисунок2.6).

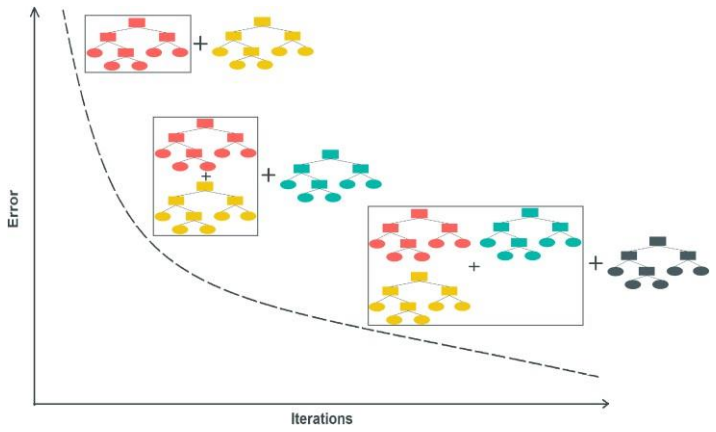


Рисунок 2.6 Основна ідея методу Gatdient Boosting

# 3. АНАЛІЗ НАБОРУ ДАНИХ ДЛЯ НАВЧАННЯ ТА ТЕСТУВАННЯ МОДЕЛЕЙ ТА МЕТРИК ОЦІНЮВАННЯ ЇХ РОБОТИ

## 3.1. Аналіз набору даних для навчання та тестування

За для побудови моделей МН я використала набір даних NSL-KDD [12] . Він є вдосконаленою версією його попередників. Він містить необхідні записи повного набору даних KDD (таблиця 3.1).

Розглянутий набір NSL-KDD має такі переваги [13]:

- немає надлишкових записів в навчальному наборі, щоб класифікатори могли створити непередбачений результат;
- достатня кількість записів є в тренувальних та тестових наборах даних (data sets), які є раціональними і розумними,це дозволяє проводити експерименти на повному наборі;
- немає дубліката записів в тестовому наборі. Він містить деякі атаки, які не присутні в навчальному наборі;
- кількість обраних записів з кожної складної групи рівнів складності обернено пропорційно частці записів у вихідному наборі даних NSL-KDD;

Data set (набір даних) містить в собі 41 атрибут, що демонструє різні особливості потоку, і клас (мітку), призначену кожному. Деталі атрибутів, а саме ім'я атрибута, їх опис та тип даних, перераховані в таблицях 3.1 та 3.2.

Контекстні ознаки на відміну від вищезгаданих ознак, які збираються безпосередньо з шлюзового вузла мережі, є предметом конструювання (feature engineering) експертами галузі і допомагають повніше відобразити ознаки несанкціонованого доступу (табл. 3.2).

Таблиця 3.1- Поля даних, що відображають основні властивості з’єднання

| Назва | Опис | Тип |
|---|---|---|
| 1. Duration | довжина тривалості з’єднання(число секунд) | чисельний |
| 2. protocol_ type | тип протоколу: tcp, udp, icmp | номінальний |
| 3. service | мережевий сервіс: http, telnet, smtp тощо | номінальний |
| 4. Flag | статус з’єднання: нормальний чи помилка | номінальний |
| 5. src_bytes | кількість переданих байтів, від джерела до пункту призначення | чисельний |
| 6. dst_bytes | Кількість переданих байтів з пункту призначення до джерела | чисельний |
| 7. Land | 1 якщо з’єднання від/до того ж host/ip | двійковий |
| 8. wrong_fragment | кількість помилкових фрагментів (wrong fragments) | чисельний |
| 9. Urgent | кількість термінових пакетів | чисельний |

Таблиця 3.2 - Контекстні ознаки кожного мережевого з'єднання

| Назва | Опис | Тип |
|---|---|---|
| 10. hot | кількість індикаторів «hot»,як наприклад, введення в системний каталог, створення програм та їх виконання | чисельний |
| 11. num_failed_logins | кількість невдалих спроб входу | чисельний |
| 12. logged_in | 1 якщо вдалий вхід в систему, інакше 0 | двійковий |
| 13. num_compromised | кількість «скомпрометованих» станів | чисельний |
| 14. root_shell | 1 якщо права root отримано, інакше 0 | двійковий |
| 15. su_attempted отримано, інакше 0 | 1 якщо права "su root" | двійковий |
| 16. num_root | кількість root-доступів | чисельний |
| 17.num_file_creations | кількість операцій створення файлів під час з'єднання | чисельний |

Продовження таблиці 3.2

| Назва | Опис | Тип |
|---|---|---|
| 18. num_shells | кількість командних рядків | чисельний |
| 19. num_access_files | кількість операцій доступу до системних файлів | чисельний |
| 20. num_outbound_cmds | кількість outbound команд протягом ftp сесії | чисельний |
| 21. is_hot_login | 1 якщо логін належить «гарячому» списку: root або admin, інакше 0 | двійковий |
| 22. is_guest_login | 1 якщо спроба гостьового входу, інакше 0 | двійковийр |

Далі йдуть параметри, пов'язані з тимчасовими характеристиками кожного мережевого з'єднання (time-based). Це ознаки обчислені методом рухомого часового вікна. Так ознаки «same host» перевіряються тільки для з'єднань протягом останніх 2 секунд які були адресовані тому ж хосту, і обчислюють статистичні характеристики пов'язані з поведінкою протоколу, служб тощо. Подібно ознаки «same service» перевіряються для з'єднань протягом останніх 2 секунд і які призначені одному сервісу.

42-й атрибут містить дані про, приналежність до одного з двох класів: нормального чи аномального. Взагалі є різні 5 класів векторів мережевих

підключень, і вони класифікуються як один нормальний клас та чотири класи атаки. 4 класи атаки додатково групуються як : Класи атак, присутні в наборі даних NSL-KDD, згруповані в чотири категорії [14,15]:

1. DOS (Denial of service)- це категорія нападу, під час якої будь-яка дія або послідовність дій приводить будь-яку частину системи, що атакується, до виходу з ладу, виснажує ресурси системи, тим самим робить її неспроможною виконувати свої функції. Причиною може бути несанкціонований доступ, затримка в обслуговуванні і так далі.

2. Probing (Зондування): метою є отримання інформації мережі або про неї (наприклад, імена і паролі користувачів) для здійснення неавторизованого доступу.

3. U2R (Unauthorized access attempt): несанкціонований доступ до локальних привілеїв суперкористувача (root) - це тип атаки, за допомогою якого зловмисник використовує звичайний обліковий запис для входу в систему жертви та намагається отримати привілеї root /адміністратора, тобто більші можливості, ніж встановлені адміністратором системи, використовуючи деяку вразливість, наприклад. атаки переповнення буфера.

4.R2L: несанкціонований доступ незареєстрованого користувача до комп'ютера з віддаленої машини. Наприклад введення пароля.

Не зважаючи на досить чітку визначеність між всіма видами мережевих атак, віднести однозначно той чи інший випадок до певного класу не можливо тому, що атака у більшості випадків – це комплекс заходів. В даній роботі для визначення мережевих атак я звузила кількість класів до двох, так як мене цікавить сам факт наявності мережевого втручання чи ні. В один кластер будуть відібрані дані ймовірно нормальної поведінки, в другий ймовірно аномальної.

Вихідна вибірка, що містить 2 класи, загалом зберігає 18894 об’єктів (пакетів) , де 10078 належать класу «нормальна поведінка, а 8816 належать класу «аномальна поведінка».

Завдяки нормалізація параметрів помітний значний приріст до виявлення аномалій. При цьому загальний відсоток правильної ідентифікації даних досягає 97%.

## 3.2. Математичні метрики оцінки якості моделі

Після етапу відбору ознак, побудови моделі і отриманню нею певних результатів у формі класів або ймовірностей, далі слід перевірити ефективності побудованих моделі.

Для цього існують спеціаліальні метрики та тестові набори даних,які обираються в залежності від типу задачі МН.

Перш ніж описати використані мною метрики, необхідно дати кілька важливих визначень.

Оцінка якості роботи класифікаторів базується на понятті Матриці Помилок (Confusion Matrix), що для задачі бінарної класифікації має наступний вигляд (таблиця 3.3):

Таблиця 3.3 – Матриця помилок

| Прогнозовані значення | True Values | y=1 | y=0 |
|---|---|---|---|
| | a(x) = 1 | True Positive (TP) | False Positive (FP) |
| | a (x) = 0 | False Negative (FN) | True Negative (TN) |

True Positive (істинно позитивне значення) – для бінарної класифікації означає наявність позитивного рішення класифікатора і що справжнє значення відповіді в оцінюючій вибірці також позитивне. False Positive (хибно-позитивне значення)–каже відповідно про наявність позитивної відповіді класифікатора і негативного в оціночної вибірці. Аналогічні випадки є і для негативних прикладів -True Negative (істинно-від'ємне значення) та False Negative(хибно- негативне значення).

Перше, що спадає на думку, коли ми хочемо виміряти якість алгоритму класифікації, це вирахувати, на скількох об'єктах ми даємо правильну відповідь, і поділити це на розмір вибірки. Ця метрика так і називається - «частка правильних відповідей» або accuracy англійською.Це найпростіша міра якості класифікатора.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

«Accuracy» обчислюється за допомогою індикаторної функції [16]:

$$accuracy = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}-1} 1(y`_i = y_i),$$

де $y`_i$ – прогнозований клас; –істинний клас; $1(y`_i = y_i)$ – індикаторна функція;

У випадках, коли різні помилки мають різну ціну, набагато краще застосовувати дві інші метрики, замість однієї якості.

Точність показує, наскільки ми можемо довіряти класифікатору, якщо він видає відповідь 1.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Друга метрика називається повнотою. Вона показує, як багато об'єктів класу 1 наш алгоритм знаходить.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Отже, точність і повнота характеризують різні сторони якості класифікатора. Чим вище точність, тим менше помилкових спрацьовувань. Чим вище повнота, тим менше помилкових пропусків.В вирішуваній мною задачі важливіше оцінити другоий пункт,що я і зробила (див. 4.2).

F1- метрика обчислюється за допомогою значень «precision» і «recall»:

$$f1\ weighted = 2 * (\frac{precision * recall}{precision + recall}),$$

F-міра дозволяє отримати більш збалансовану характеристику моделі, ніж три метрики, розглянуті вище. Однак, оскільки F-міра не враховує TN, ця метрика також може дати зміщену оцінку.

Часто результат роботи алгоритму на фіксованій тестової вибірці візуалізують за допомогою ROC-кривої, а якість оцінюють як площу під цією кривої - AUC (AUC = area under the curve).

ROC-крива - представляє із себе графічну характеристику якості бінарного класифікатора, залежність частки вірних позитивних класифікацій від частки помилкових позитивних класифікацій при варіюванні порога вирішального правила, тобто лінію від (0,0) до (1,1) в координатах True Positive Rate (TPR) і False Positive Rate, де TPR=повноті,а FPR - показує, яку частку з об'єктів класу, що дають негативну відповідь, алгоритм передбачив невірно.

$$TPR = \frac{TP}{TP+FN};$$

$$FPR = \frac{FP}{FP+TN};$$

Дана крива ілюструє чутливість класифікатору, показуючи скільки правильно класифікованих об'єктів можна отримати, дозволяючи більше й більше випадків FP.

ROC-крива монотонно не зменшується. Чим вище лежить крива, тим краще якість класифікації.

Площа під ROC-кривої теж є хорошою метрикою якості, вона так і називається AUC-ROC або площа під ROC-кривою.

У роботі [18] говориться про те, що AUC– дуже об'єктивна і вона не вимагає особливого вкладу від користувача. Це агрегована характеристика якості класифікації, що не залежить від співвідношення цін помилок. Чим більше значення AUC, тим «краще» модель класифікації. AUC видає результати у періоді від 0 до 1. Модель, чиї прогнози 100% помилкові, має AUC рівну 0.0, а модель з 100% вірними прогнозами має AUC рівну 1.0.

$$AUC = \int_{x=0}^{1} TPR\left(FPR^{-1}(x)\right)dx =$$

$$= \int_{+\infty}^{-\infty} TPR(T)FPR'(T)dT = \iint_{-\infty}^{\infty} I(T' > T)f_1(T')f_0(T)dT'dT = P(X_1 > X_0)$$

де T –пороговий змінний параметр; $X_1$ – оцінка TPR; $X_0$ – оцінка FPR;

$f_1, f_0$ - щільність ймовірності[18].

# 4. СТВОРЕННЯ СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

## 4.1. Протокол дослідження моделей

В даній дипломній роботі було реалізовано шість моделей МН (рисунок 4.7, 4.9- 4.13) та отримано метрики точності кожної з моделей(таблиця4.1).

Моделі створювались в середовищі Jupyter Notebook, з використанням бібліотеки МН Scikit-learn (рисунок 4.1). Початковий набір даних попередньо був розділений на тренувальний та тестовий (рисунок 4.2). В тренувальний потрапили всі дані, а в тестовий лише частина.

```
#модулі матречних обчислень та маніпуляції наборами даних
import pandas as pd
import numpy as np
#модулі побудови графіків
import seaborn as  sns
from matplotlib import pyplot
import matplotlib.pyplot as plt
#модулі МН та оцінювання моделей
from sklearn.metrics import
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import mean_squared_error as mse
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import LinearSVC
from sklearn.ensemble import VotingClassifier
from sklearn.feature_selection import RFECV
from sklearn.metrics import roc_curve
from sklearn import tree
from string import ascii_letters
```

Рисунок 4.1- Модулі

```
#Зчитуємо тренувальний та естовий набори даних з файла
df_train= pd.read_excel(r"C:\Users\Александр\Desktop\train.xlsx")
df_test=pd.read_excel(r"C:\Users\Александр\Desktop\test.xlsx")
```

Рисунок 4.2- Завантаження наборів даних

Переглянемо перщі 5 записів обох вибірок( рисунок 4.3)

```
#перщі 5 записів тренувальної вибірки
df_train.tail()
```

| | duration | protocol_type | service | flag | src_bytes | dst_bytes | land | wrong_fragme |
|---|---|---|---|---|---|---|---|---|
| **18889** | 0 | udp | domain_u | SF | 45 | 81 | 0 | |
| **18890** | 0 | tcp | login | S0 | 0 | 0 | 0 | |
| **18891** | 0 | tcp | http | S0 | 0 | 0 | 0 | |
| **18892** | 0 | tcp | http | SF | 216 | 2361 | 0 | |
| **18893** | 0 | icmp | ecr_i | SF | 1032 | 0 | 0 | |

5 rows × 42 columns

```
#перщі 5 записів тестової вибірки
df_test.tail()
```

| | duration | protocol_type | service | flag | src_bytes | dst_bytes | land | wrong_fragmer |
|---|---|---|---|---|---|---|---|---|
| **6293** | 0 | tcp | exec | RSTO | 0 | 0 | 0 | |
| **6294** | 0 | tcp | ftp_data | SF | 334 | 0 | 0 | |
| **6295** | 0 | tcp | private | REJ | 0 | 0 | 0 | |
| **6296** | 0 | tcp | nnsp | S0 | 0 | 0 | 0 | |
| **6297** | 0 | tcp | finger | S0 | 0 | 0 | 0 | |

5 rows × 41 columns

Рисунок 4.3- Перші 5 записів вибірок

Виведемо на друк кількість записів з пункту class (рисунок 4.4):

```
counts = df_train["class"].value_counts()
print(counts)
```

```
normal     10078
anomaly     8816
Name: class, dtype: int64
```

Рисунок 4.4- Кількість об’єктів в пункті class

Графічно зобразимо співвідношення нормального та аномального класів(рисунок 4.5).

```
#Show the counts of observations in each categorical bin of train data
plt.figure(figsize=(8,5))
sns.set(font_scale=1.3)
ax = sns.countplot(x="class", data=df_train,linewidth=3,palette="Accent_
```

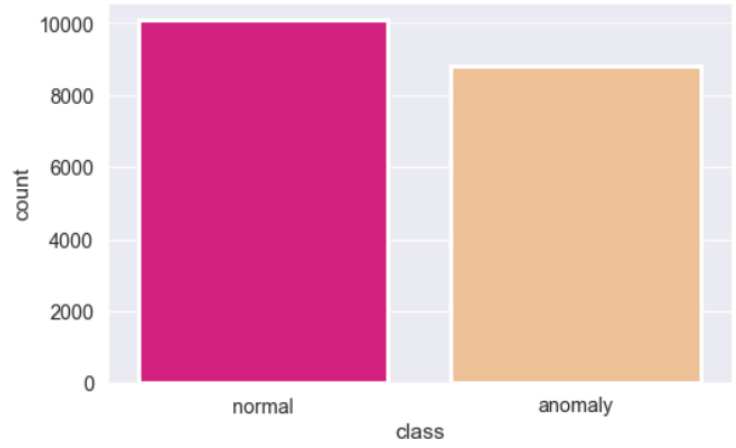


Рисунок 4.5- Співвідношення класів

Кодуємо та переформатовуємо символьні значення в бінарну матрицю. Кодуємо категорійні число значення за схемою one of k (рисунок 4.6).

```
df_train_2['service'].unique()
```

```
array(['ftp_data', 'other', 'private', 'http', 'remote_job', 'name',
       'netbios_ns', 'eco_i', 'mtp', 'telnet', 'finger', 'domain_u',
       'supdup', 'uucp_path', 'Z39_50', 'smtp', 'csnet_ns', 'uucp',
       'netbios_dgm', 'urp_i', 'auth', 'domain', 'ftp', 'bgp', 'ldap',
       'ecr_i', 'gopher', 'vmnet', 'systat', 'http_443', 'efs', 'whois',
       'imap4', 'iso_tsap', 'echo', 'klogin', 'link', 'sunrpc', 'login',
       'kshell', 'sql_net', 'time', 'hostnames', 'exec', 'ntp_u',
       'discard', 'nntp', 'courier', 'ctf', 'ssh', 'daytime', 'shell',
       'netstat', 'pop_3', 'nnsp', 'IRC', 'pop_2', 'printer', 'tim_i',
       'pm_dump', 'red_i', 'netbios_ssn', 'rje', 'X11', 'urh_i',
       'http_8001'], dtype=object)
```

```
df_train_2['flag'].unique()
```

```
array(['SF', 'S0', 'REJ', 'RSTR', 'SH', 'RSTO', 'S1', 'RSTOS0', 'S3',
       'S2', 'OTH'], dtype=object)
```

Рисунок 4.6- Переформатування

Навчання та тестування моделей класифікаторів на основі методів LR,k-NN,SVM,DT,RF,GB (рисунки 4.7, 4.9- 4.13):

```
#Train LR model
LR.fit(X_train,Y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
pickle.dump(LR, open('LR_model_SA.sav', 'wb'))
#load model from file
```

```
LR_model_SA = pickle.load(open('LR_model_SA.sav', 'rb'))
#result = KNN_loaded_model.score(X_test, Y_test)
```

```
y_pred = LR_model_SA.predict(X_test)
#predict probabilities
probs = LR_model_SA.predict_proba(X_test)
probs = probs[:,1]
# roc_auc_score (y_test,probs)
auc = roc_auc_score(Y_test, probs)
#calc roc curve
fpr,tpr,thresholds = roc_curve(Y_test, probs)
pyplot.plot([0,1], [0, 1],'b--')
#plot the roc curve for the model
pyplot.title("ROC and AUC for LR model (SA)")
plt.xlabel('FPR')
plt.ylabel('TPR')
pyplot.plot(fpr, tpr, marker = '.')
pyplot.show()
```

Рисунок 4.7- Навчання та тестування моделі на основі метода LR

Далі оцінюємо роботу моделі за допомогою метрик та визначаємо загальне число помилок

```
print("Довжина тестової вибірки: %s" % Len(y_test[y_test!=y_predicted]))
```

```
Кількість записів які передбачено що є нормальними а насправді є
вторгненнями: 78
```

Рисунок 4.8- Загальне число помилок

Система пропустила деякі загрози. Надалі більш детальна оцінка не лише даної моделі, а всіх разом буде далі в матриці з показниками метрик.

```
#Train KNN model
KNN.fit(X_train,Y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

```
#save model to file
pickle.dump(KNN, open('KNN_model_SA.sav', 'wb'))#KNN_model_operscionalizatior
```

```
KNN_model_SA = pickle.load(open('KNN_model_SA.sav', 'rb'))
#result = KNN_loaded_model.score(X_test, Y_test)
```

```
y_pred = KNN_model_SA.predict(X_test)
#predict probabilities
probs = KNN_model_SA.predict_proba(X_test)
probs = probs[:,1]
# roc_auc_score (Y_test,probs)
auc = roc_auc_score(Y_test, probs)
#calc roc curve
fpr,tpr,thresholds = roc_curve(Y_test, probs)
pyplot.plot([0,1], [0, 1],'b--')
#plot the roc curve for the model
pyplot.title("ROC and AUC for KNN model (SA)")
plt.xlabel('FPR')
plt.ylabel('TPR')
pyplot.plot(fpr, tpr, marker = '.')
pyplot.show()
```

Рисунок 4.9- Навчання та тестування моделі на основі метода k-NN

```
#Train SVM model
SVM.fit(X_train,Y_train)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=True, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

```
pickle.dump(SVM, open('SVM_model_SA.sav', 'wb'))
#load model from file
```

```
SVM_model_SA = pickle.load(open('SVM_model_SA.sav', 'rb'))
#result = KNN_loaded_model.score(X_test, Y_test)
```

```
y_pred = SVM_model_SA.predict(X_test)
#predict probabilities
probs = SVM_model_SA.predict_proba(X_test)
probs = probs[:,1]
# roc_auc_score (y_test,probs)
auc = roc_auc_score(Y_test, probs)
#calc roc curve
fpr,tpr,thresholds = roc_curve(Y_test, probs)
pyplot.plot([0,1], [0, 1],'b--')
#plot the roc curve for the model
pyplot.title("ROC and AUC for SVM model (SA)")
plt.xlabel('FPR')
plt.ylabel('TPR')
pyplot.plot(fpr, tpr, marker = '.')
pyplot.show()
```

Рисунок 4.10- Навчання та тестування моделі на основі метода SVM

```
#Train DecisionTree model
DecisionTree.fit(X_train,Y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

```
pickle.dump(DecisionTree, open('DecisionTree_model_SA.sav', 'wb'))
#load model from file
```

```
DecisionTree_model_SA = pickle.load(open('DecisionTree_model_SA.sav', 'rb'))
#result = KNN_loaded_model.score(X_test, Y_test)
```

```
y_pred = DecisionTree_model_SA.predict(X_test)
#predict probabilities
probs = DecisionTree_model_SA.predict_proba(X_test)
probs = probs[:,1]
# roc_auc_score (Y_test,probs)
auc = roc_auc_score(Y_test, probs)
#calc roc curve
fpr,tpr,thresholds = roc_curve(Y_test, probs)
pyplot.plot([0,1], [0, 1],'b--')
#plot the roc curve for the model
pyplot.title("ROC and AUC for DecisionTree model (SA)")
plt.xlabel('FPR')
plt.ylabel('TPR')
pyplot.plot(fpr, tpr, marker = '.')
pyplot.show()
```

Рисунок 4.11- Навчання та тестування моделі на основі метода DT

```
#Train DecisionTree model
RandomForest.fit(X_train,Y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=N
one,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
pickle.dump(RandomForest, open('RandomForest_model_SA.sav', 'wb'))
#load model from file
```

```
RandomForest_model_SA = pickle.load(open('RandomForest_model_SA.sav', 'rb'))
#result = KNN_loaded_model.score(X_test, Y_test)
```

```
y_pred = RandomForest_model_SA.predict(X_test)
#predict probabilities
probs = RandomForest_model_SA.predict_proba(X_test)
probs = probs[:,1]
# roc_auc_score (Y_test,probs)
auc = roc_auc_score(Y_test, probs)
#calc roc curve
fpr,tpr,thresholds = roc_curve(Y_test, probs)
pyplot.plot([0,1], [0, 1],'b--')
#plot the roc curve for the model
pyplot.title("ROC and AUC for RandomForest model (SA)")
plt.xlabel('FPR')
plt.ylabel('TPR')
pyplot.plot(fpr, tpr, marker = '.')
pyplot.show()
```

Рисунок 4.12- Навчання та тестування моделі на основі метода RF

```
#Train GradientBoosting model
GradientBoosting.fit(X_train,Y_train)
```

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='auto',
                           random_state=None, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

```
pickle.dump(GradientBoosting, open('GradientBoosting_model_SA.sav', 'wb'))
#load model from file
```

```
GradientBoosting_model_SA = pickle.load(open('GradientBoosting_model_SA.sav', 'rb'))
#result = KNN_loaded_model.score(X_test, Y_test)
```

```
y_pred = GradientBoosting_model_SA.predict(X_test)
#predict probabilities
probs = GradientBoosting_model_SA.predict_proba(X_test)
probs = probs[:,1]
# roc_auc_score (Y_test,probs)
auc = roc_auc_score(Y_test, probs)
#calc roc curve
fpr,tpr,thresholds = roc_curve(Y_test, probs)
pyplot.plot([0,1], [0, 1],'b--')
#plot the roc curve for the model
pyplot.title("ROC and AUC for GradientBoosting model (SA)")
plt.xlabel('FPR')
plt.ylabel('TPR')
pyplot.plot(fpr, tpr, marker = '.')
pyplot.show()
```

Рисунок 4.13- Навчання та тестування моделі на основі метода GB

## 4.2. Аналіз отриманих результатів метрик точності моделей

Експерименти були проведені в ОС Windows 10 з процесором Core i5, 6.0 Гб ОЗУ. Оцінка інформативності ознак проводилася в командній оболонкі для інтерактивних обчислень на навчальному і тестовому вибірках даних бази KDD (рисунок 4.14)(таблиця 4.1)

```
data = {'fit_time':[GradientBoosting_for_time],
        'score_time':[GradientBoosting_score_time],
        'test_accuracy':[GradientBoosting_test_accuracy],
        'test_precision_macro':[GradientBoosting_test_precision_macro],
        'test_recall_macro':[GradientBoosting_test_recall_macro],
        'test_f1_weighted':[GradientBoosting_test_f1_weighted],
        'test_roc_auc':[GradientBoosting_test_roc_auc]}
df = pd.DataFrame(data, index =['GradientBoosting'])
df
df.to_excel("scores_GradientBoosting(T).xlsx")
```

```
data = {'fit_time':[LR_for_time, KNN_for_time, DecisionTree_for_time],
        'score_time':[LR_score_time, KNN_score_time, DecisionTree_score_time],
        'test_accuracy':[LR_test_accuracy, KNN_test_accuracy, DecisionTree_test_accuracy],
        'test_precision_macro':[LR_test_precision_macro, KNN_test_precision_macro, DecisionTree_test_precision_macro],
        'test_recall_macro':[LR_test_recall_macro, KNN_test_recall_macro, DecisionTree_test_recall_macro],
        'test_f1_weighted':[LR_test_f1_weighted, KNN_test_f1_weighted, DecisionTree_test_f1_weighted],
        'test_roc_auc':[LR_test_roc_auc, KNN_test_roc_auc, DecisionTree_test_roc_auc]}
df = pd.DataFrame(data, index =['LR', 'KNN','SVM','DecisionTree'])
df.to_excel("scoress_LR_KNN_DecisionTree_HYDROLASE.xlsx")
```

Рисунок 4.14-Програмування метрик

Таблиця 4.1-Метрики точності алгоритмів для моделі класифікації

| Column1 | fit_time | score_time | test_accuracy | test_f1_weighted | test_recall_macro | test_roc_auc | Сумма test_precision_macro |
|---|---|---|---|---|---|---|---|
| SVM | 309,15 | 17,00 | 0,95 | 0,95 | 0,94 | 0,99 | 0,95 |
| GradientBoosting | 1,93 | 0,02 | 0,99 | 0,99 | 0,99 | 1,00 | 0,99 |
| LR | 0,38 | 0,01 | 0,89 | 0,89 | 0,88 | 0,95 | 0,89 |
| KNN | 0,10 | 1,84 | 0,99 | 0,99 | 0,99 | 1,00 | 0,99 |
| RandomForest | 0,10 | 0,02 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |
| DecisionTree | 0,09 | 0,01 | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |

В результаті помилка алгоритмів склала не більше 0,11% при кількості дерев рівному 30 і значенні OOB рівному 0,03%, що показує хорошу роботу підходу (рисунок 4.15 - 4.17).

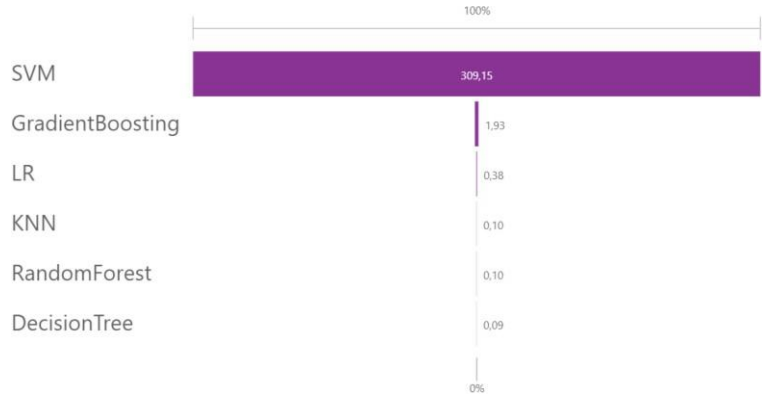


Рисунок 4.15 Співвідношення метрики «fit time»

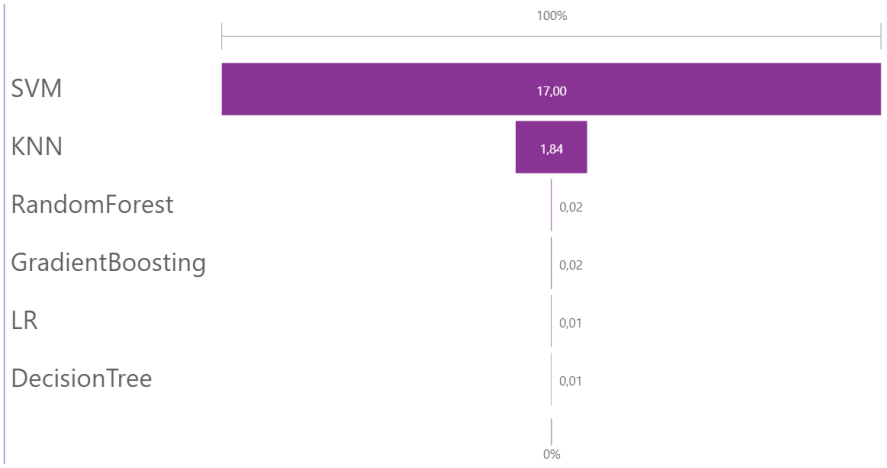


Рисунок 4.16 Співвідношення метрики «score time»

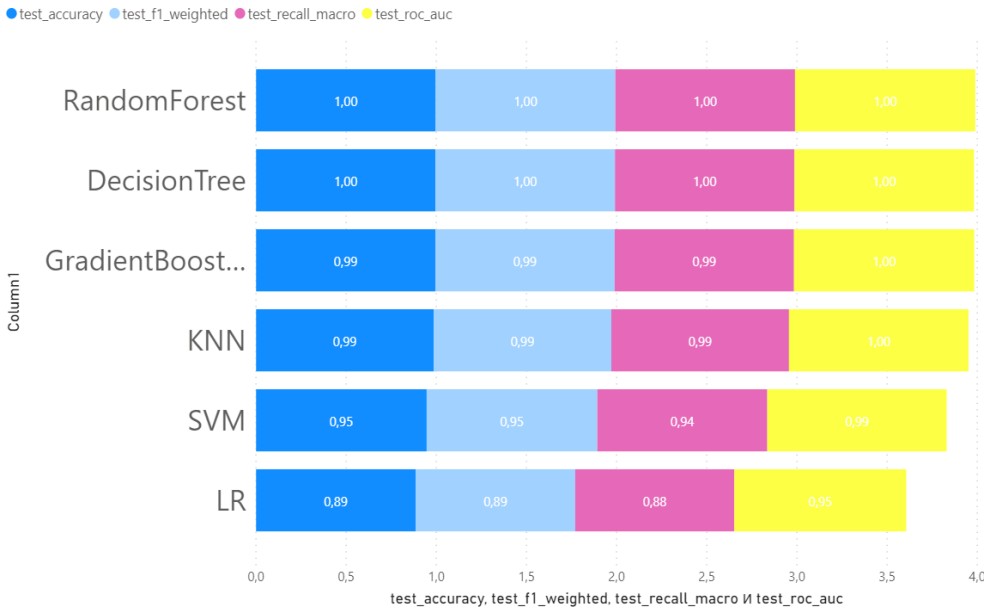


Рисунок 4.17 -Співвідношення метрик «accuracy», «f1 weighted», «recall», «roc-auc» для моделей класифікації

Грунтуючись на нашому порівняльному аналізі , результати у таблиці 4.1 та результатах, зображених на рисунках 4.15, 4.16, 4.17, можна зробити

висновок про те, що кращими метриками для класифікації мережевих втручань в тестовому наборі даних є RF та DT. Ці алгоритми не лише швидкі, а й максимально точні, вони чудово навчились на тренувальному наборі.

У порівнянні з іншими алгоритмами довше всіх узагальнюється на тестових даних алгоритм SVM. Та при достатньо великому часі узагальнення, цей алгоритм показав не найгірші значення відповідно до інших метрик, а саме «accuracy», «precision», «f1 weighted» «recall_macro» та «roc-auc». Найменш точним є алгоритм LR, хоча й він доволі швидкий.

Алгоритм GB має прийнятний час узагальнення та доволі велику точність результату. Проте, варто зазначити, що алгоритм LR узагальнюється трішки швидше.

Застосування ансамблів моделей може не перевершувати за якістю прогнозу окремі алгоритми. Так виявилось в нашому випадку, найбільш точним та оптимальним за часом узагальнення є модель, з використанням алгоритму DT, так як він, хоч лише й на 0,01,та все ж обійшов модель з використанням алгоритму ансамблювання RF в показаннях метрики «score_time» .

## 4.3. Аналіз роботи алгоритмів відповідно до ROC та AUC

На даних графіках (рисунок4.4) передаємо в якості x-координат FPR, в якості y- координат TPR (див. 3.2).

З рисунку 4.18 видно, що найкраще класифікують моделі з використанням алгоритмів GB,RF та DT. Найгіршу ж роботу демонструє алгоритм LR), для якого AUC має найменше значення.

```
# Compute macro-average ROC curve and ROC area
n_classes=2
i=0
# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure()
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to multi-class')
plt.legend(loc="lower right")
plt.show()
```

Рисунок 4.17- Створення,налаштування та візуалізація метрики ROC AUC

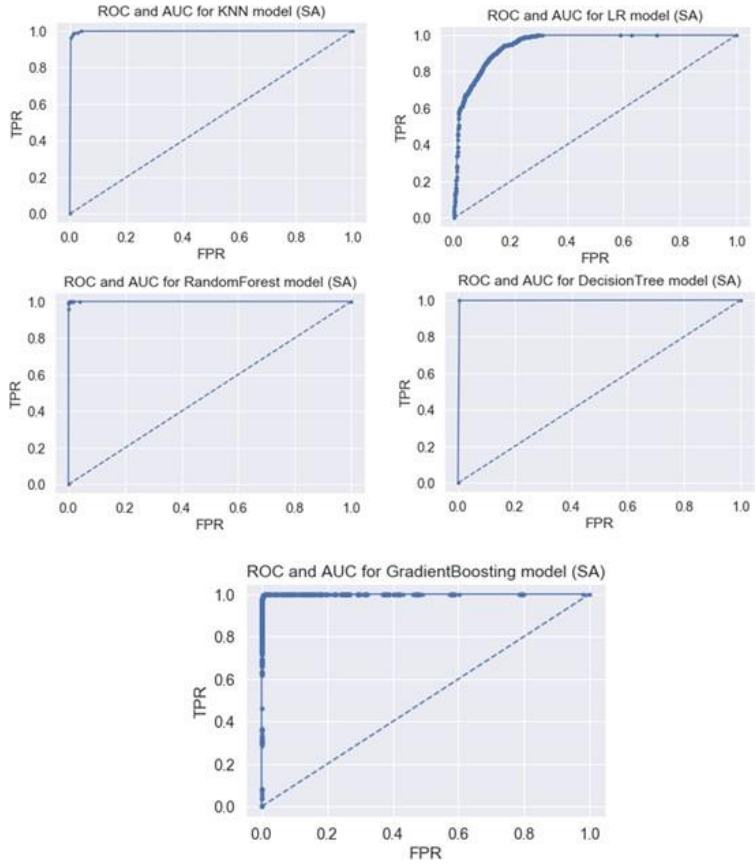


Рисунок 4.18 Графіки ROC-кривої роботи алгоритмів для моделей класифікації

## 4.4. Технології, використані для створення моделей машинного навчання

Pandas [18]- бібліотека, яка робить Python потужним інструментом для аналізу та обробки даних. Вона забезпечує швидку, чітку та пластичну структуру даних, дає можливість будувати зведені таблиці, виконувати угруповання, надає зручний доступ до табличних даних, а при наявності пакета matplotlib дає можливість малювати графіки на отриманих наборах даних. Розроблена для спрощення роботи зі структурованими даними та даними часових рядів.

NumPy [19]- це бібліотека мови Python. Вона забезпечує підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих (і дуже швидких) математичних функцій для операцій з цими масивами.

«Matplotlib» [20]- це всебічна бібліотека для візуалізації даних двовимірної (2D) графікою (3D графіка також підтримується) на Python, за допомогою якої можна створювати високоякісні малюнки різних форматів, що є важливим в машинному навчанні, оскільки це допомагає краще пояснити отримані результати.

Seaborn- це більш високорівнева API (application programming interface) на базі бібліотеки matplotlib. Seaborn містить більш відповідні за умовами та налаштуваннями, які визначені відпочатку настройки оформлення графіків. Якщо додати в код import seaborn, то малюнки стануть набагато привабливішими. До того ж в даній бібліотеці є досить складні типи візуалізації, які в matplotlib вимагали більшої кількість коду.

CountVectorizer перетворює вхідний текст в матрицю, значеннями якої є кількості входження даного ключа (слова) в текст.

Scikit-learn [18] – бібліотека, що орієнтована на моделювання даних, через систематичний інтерфейс в Python вона забезпечує ряд контрольованих та непідконтрольних алгоритмів навчання.

Sklearn.metrics- це модуль, що надає готову реалізацію більшості метрик, що використовуються в задачах класифікації для оцінювання якості отриманих моделей.

Модулі, які надають можливість використання обраних мною алгоритмів( рисунок 4.19).

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
```

Рисунок 4.19- Sklearn.metrics

## 4.5. Вбудовування моделі машинного навчання в веб-додаток

Прийоми МН не обмежуються автономними додатками і аналізом. Вони можуть виступати в якості прогнозуючих механізмів веб-служб[21].

Процес вбудовування моделі можна розділити на такі основні етапи:

- збереження поточного стану навченої моделі МН;
- використання баз даних SQLite для сховищ даних;
- розробка веб-додатку із застосуванням популярної веб-інфраструктури Flask;

- розгортання програми МО на публічному веб-сервері.

Почати слід з пункту збереження поточного стану навченої моделі, адже навчати модель заново після кожного закриття інтерпретатора Python або перезавантаження веб-додатку це марна трата часу. Одним з варіантів забезпечення сталості моделей є модуль pickle для Python. Він реалізує серіаліацію і десеріаліацію структур об'єктів Python з метою стиснення байт-коду, щоб можна було зберегти класифікатор в його поточному стані і знову завантажити його, коли потрібно класифікувати нові зразки, не змушуючи модель вчитися на навчальних даних ще раз (рисунок 4.20)[22].

```
import pickle
import os
   dest = os.path.join('trafficclassifier', 'pkl_objects')
if not os.path.exists(dest):
   os.makedirs(dest) pickle.dump(stop,
open(os.path.join(dest, 'classifier.pkl'), 'wb'),protocol=4)
```

Рисунок 4.20- Модуль pickle

У наведеному вище коді створюється каталог trafficclassifier, де пізніше будемо зберігати дані для нашого веб-додатку. Усередині каталогу trafficclassifier ми створюємо підкаталог pkl_objects, щоб зберігати на локальному диску серіалізовані об'єкти Python. За допомогою методу dump модуля pickle ми потім серіалізуємо навчену класифікаційну модель .

Метод dump приймає в своєму першому аргументі об'єкт, який слід законсервувати, а в другому аргументі - відкритий файловий об'єкт, куди буде записуватися об'єкт Python.

За допомогою аргументу wb всередині функції open ми відкриваємо файл в двійковому режимі для консервації і встановлюємо protocol = 4, щоб вибрати самий останній і ефективний протокол консервації.

Створюємо файл сценарію Python, з якого імпортується векторизатор в поточний сеанс Python і він зберіється у файлі vectorizer.py всередині каталогу trafficclassifier. Після успішного завантаження vectorizer і розконсервування класифікатора ми можемо використовувати ці моделі для вироблення прогнозів. Оскільки наш класифікатор повертає мітки класів як цілі числа, ми визначили простий словник Python для відображення таких цілих чисел на їх відносини (рисунок 4.21).

```
import numpy as np
label = {0:'anomaly', 1:'normal'}
example = [1.xlsx]
X = vect.transform(example)
print('Прогноз: %s\nВероятность: %.2f%%' %\(label[clf.predict(X)[0]],np.max(clf.predict_proba(X))*100))
Прогноз: normal
Вероятность: 91.56%
```

Рисунок 4.21- Словник Python

Потім ми застосували HashingVectorizer для трансформації документа в вектор значень X. Нарешті, ми використовували метод predict класифікатора на основі дерева рішень, так як даний алгорит показав найкращу роботу, для прогнозування мітки класу, а також метод predict_проба для повернення відповідної ймовірності прогнозу.

Далі для поновленню класифікаційної моделі використовується наступний пункт. Налаштування простої бази даних SQLite для збору файлів чи об’єктів, які класифікували користувачі веб-додатку.

SQLite - це механізм баз даних SQL з відкритим кодом, що не вимагає для своєї роботи окремого сервера, що робить його ідеальним варіантом при розробці невеликих проектів і простих веб-додатків.

Стандартна бібліотека Python пропонує API-інтерфейс sqlite3, який дозволяє працювати з базами даних SQLite. Виконавши наступний код, ми

створимо всередині каталогу trafficclassifier нову базу даних SQLite (рисунок 4.22).

```
import sqlite3
import os
if os.path.exists('files.sqlite'): os.remove('files.sqlite')
conn = sqlite3.connect('files.sqlite')
c = conn.cursor()
c.execute('CREATE TABLE files_db'\ ' (files TEXT, class INTEGER, date TEXT)')
conn.commit()
conn.close()
```

Рисунок 4.22- Створення бази даних SQLite

У коді ми створюємо підключення (conn) до файлу бази даних SQLite, викликаючи метод connect з бібліотеки sqlite3, який створює в каталозі trafficclassifier файл бази даних files.sqlite, якщо він поки не існує.

Потім за допомогою методу cursor ми створюємо курсор, який дасть можливість проходити по записах бази даних, використовуючи універсальний синтаксис SQL. Далі із застосуванням першого виклику execute ми створюємо нову таблицю бази даних files_db, яку будемо використовувати для збереження і подальшого доступу до записів. У таблиці files_db ми створюємо три стовпці: files, class і date. Вони будуть застосовуватися для зберігання прикладів об‘єктів і відповідних позначок класів.

На закінчення ми викликаємо метод commit для збереження змін, внесених в базу даних, і закриваємо з'єднання за допомогою методу close.

Використовується інфраструктура Flask для розробки веб-додатку, оскільки вона написана на мові Python, тому постачає зручний інтерфейс для вбудовування існуючого коду Python[23].

Файл app.py містить основний код, який буде виконуватися інтерпретатором Python для запуску веб-додатки Flask. В каталозі templates

інфраструктура Flask буде шукати статичні HTML-файли для візуалізації в веб-браузері. вміст файлу app.py зображений на рисунках рисунок 4. 23 та 4.24.

Ми запускаємо додаток як одиночний модуль. Відповідно, ми ініціалізуємо новий екземпляр Flask з аргументом__name__, щоб інфраструктурі Flask було відомо, що підкаталог HTMLшаблонів (templates) можна відшукати в тому ж каталозі, де знаходиться додаток.

2. Потім ми застосовуємо декоратор маршруту (@ app.route ( '/')), щоб вказати URL, який повинен ініціювати виконання функції index.

3. Далі функція index просто візуалізує HTML-файл model_app. html, що знаходиться в підкаталозі templates.

4. Нарешті, ми використовуємо функцію run для запуску програми на сервері, коли цей сценарій безпосередньо виконується інтерпретатором Python, що гарантується оператором if з умовою name == ' main  '.

```
from flask import Flask, render_template, request
from wtforms import Form, TextAreaField, validators
import pickle
import sqlite3
import os
import numpy as np
# імпорт HashingVectorizer з локального каталога
from vectorizer import vect
app = Flask(__name__)
######## Підготовка класифікатора
cur_dir = os.path.dirname(__file__)
clf = pickle.load(open(os.path.join(cur_dir,'pkl_objects', 'classifier.pkl'), 'rb'))
db = os.path.join(cur_dir, 'reviews.sqlite')
def classify(document):
 label = {0: 'негативный', 1: 'позитивный'}
 X = vect.transform([document])
 y = clf.predict(X)[0]
 proba = np.max(clf.predict_proba(X))
 return label[y], proba
def train(document, y):
 X = vect.transform([document])
 clf.partial_fit(X, [y])
def sqlite_entry(path, document, y):
 conn = sqlite3.connect(path)
 c = conn.cursor()
 c.execute("INSERT INTO files_db (files, sentiment, date)"\
 " VALUES (?, ?, DATETIME('now'))", (document, y))
 conn.commit()
 conn.close()
```

Рисунок 4.23- Імпорт модів і об’єктів Python

На рисунку 4.24 імпортовано HashingVectorizer і розконсервовано класифікатор на основі дерева рішєнь. Потім визначається функція classify для повернення спрогнозованої мітки класу і ймовірності прогнозу, що відносяться до заданого документу, формату excel. Функція train може застосовуватися для поновлення класифікатора за умови, що надані документ і мітка класу. З використанням функції sqlite_entry ми можемо зберігати в базі даних введений або завантажений користувачем набір даних з міткою класів і відміткою часу в реєстраційних цілях. Зверніть увагу, що при перезапуску веб-додатку об'єкт clf буде скидатися у вихідний законсервоване стан.

Далі визначається клас ReviewForm, що створює екземпляр DataAreaField, який буде візуалізовано в файлі шаблону reviewform.html (цільова сторінка веб-додатку). У свою чергу це візуалізується функцією index

Усередині функції results ми витягаємо вміст введеної або завантаженої веб-форми і передаємо його класифікатору для вироблення прогнозу класу, який потім відображається в шаблоні results.html (рисунок 4.25).

```
class ReviewForm(Form):
 detectionanomalies = TextAreaField('',
 [validators.DataRequired()])
@app.route('/')
def index():
 form = ReviewForm(request.form)
 return render_template('trafficform.html', form=form)
@app.route('/results', methods=['POST'])
def results():
 form = TrafficForm(request.form)
 if request.method == 'POST' and form.validate():
 review = request.form['detectionanomalie']
 y, proba = classify(review)
 return render_template('results.html',content=data,prediction=y,probability=round(proba*100, 2))
 return render_template('trafficform.html', form=form)
@app.route('/Save results', methods=['POST'])
 prediction = request.form['prediction']
 inv_label = {'negative': 0, 'positive': 1}
 y = inv_label[prediction]
 train(review, y)
 sqlite_entry(db, review, y)
 return render_template('done.html')
if __name__ == '__main__':
 app.run(debug=True)
```

Рисунок 4.24 – Визначення класу ReviewForm

Після того, як веб-додаток протестовано локально, його можна розгортати на публічному веб-сервері[24]. В даному проєкті використовується служба веб-хостингу PythonAnywhere[25].

## 4.6. Користувацький інтерфейс

При переході у веб-додаток спочатку відображається вікно вводу даних (рисунок 4.26) для класифікації. З лівого боку розміщений зафіксований sidebar,який містить два елементи. При натисненні на нижній,на якому зображений монітор, відкриється вікно завантаження файлів для класифікації (рисунок 4.29). Верхній елемент буде відкривати вікно вводу даних, якщо воно вже відкрите, то нічого не зміниться, введені данні залишаться.


| | protocol_type | service | flag | src_bytes | dst_bytes | land | wrong_fragm |
|---|---|---|---|---|---|---|---|
| 1 | tcp | http | SF | 342 | 9542 | 0 | |
| 2 | icmp | ecr_i | SF | 1032 | 0 | 0 | |
| 3 | tcp | http | SF | 264 | 19932 | 0 | |
| 4 | icmp | eco_i | SF | 8 | 0 | 0 | |
| 5 | tcp | http | SF | 300 | 413 | 0 | |
| 6 | tcp | http | SF | 224 | 994 | 0 | |
| 7 | tcp | finger | S0 | 0 | 0 | 0 | |
| 8 | tcp | private | S0 | 0 | 0 | 0 | |
| 9 | tcp | ftp | SF | 1441 | 4152 | 0 | |
| 10 | udp | domain_u | SF | 46 | 75 | 0 | |
| 11 | udp | other | SF | 516 | 4 | 0 | |
| 12 | tcp | private | S0 | 0 | 0 | 0 | |
| 13 | tcp | private | REJ | 0 | 0 | 0 | |
| 14 | tcp | nnsp | S0 | 0 | 0 | 0 | |
| 15 | tcp | private | S0 | 0 | 0 | 0 | |
| 16 | tcp | http | SF | 232 | 16070 | 0 | |
| 17 | tcp | private | S0 | 0 | 0 | 0 | |
| 18 | tcp | smtp | SF | 2308 | 336 | 0 | |
| 19 | | | | | | | |

Рисунок 4.26- Вікно вводу даних для класифікації

Введення даних не обмежене. Після того, як користувач введе всі бажані дані, для того щоб їх класифікувати він має натиснути кнопку Auto classification, що знаходиться справа зверху. При натисненні off зміниться

на on та з’ явиться символ завантеження (рисунок 4.27).Він повідомляє про те що класифікація відбувається,швидкість її виконнання залежить від кількості введених даних. Під час завершення класифікації символ завантаження зникне, прогрузиться поле complete і коли воно додівнюватиме 100% справа від нього з’являться дві кнопки(рисунок 4.28).


Рисунок 4.27- Процес класифікації

При натисненні кнопки save results класифіковані файли буде завантажено на компьютер користувача в файлі формату excel. При натисненні кнопки new, попередньо введені данні видаляться та можна буде вводити нові.


Рисунок 4.28- Завершення класифікації

Якщо користувач не бажає вводити дані в ручну він може перейти в вікно завантаження файлу для класифікації. Для того щоб завантажити файл,слід перемістити його в блакитну зону,що виділена пунктиром.

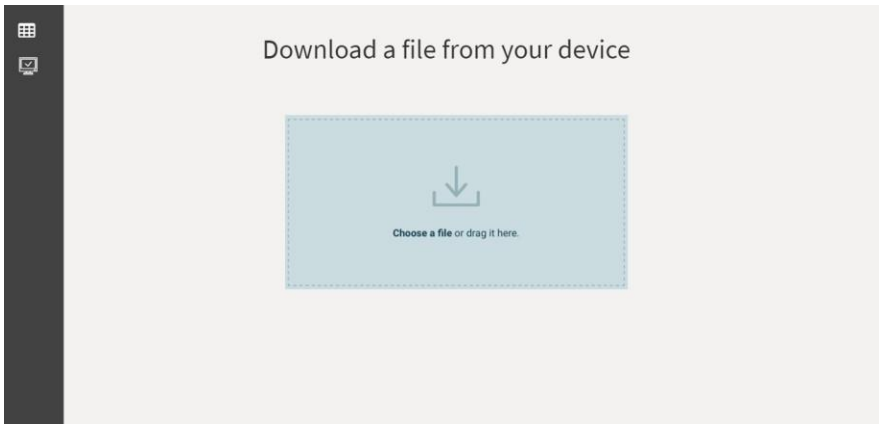


Рисунок 4.29- Вікно завантаження даних для класифікації

Про те, що файл завантажується будуть сдівчити зміни зображені на рисунку 4.30. Швидкість завантаження залежить від розміру завантажуваного файлу. Та в будь якому разі загрузка доволі швидка.

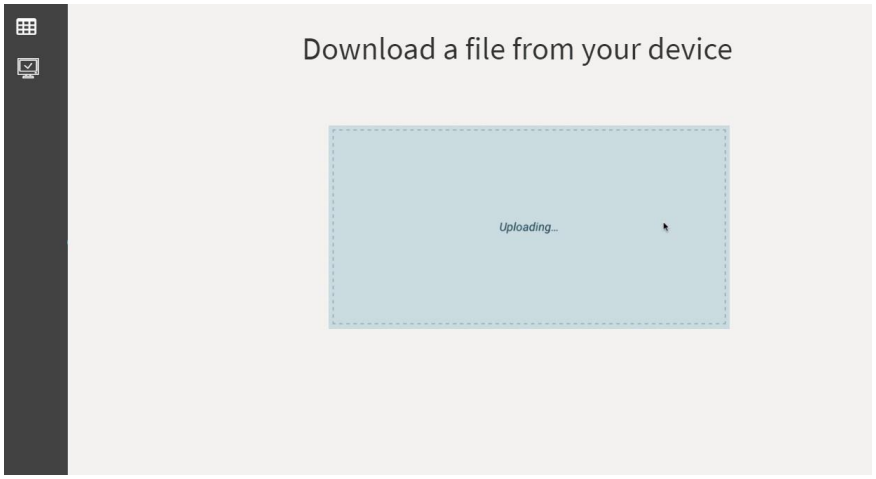


Рисунок 4.30- Завантаження даних

Після завантаження поточного файлу (рисунок 4.31), можна обрати ще й інші або одразу перемістити у вікно декілька файлів.

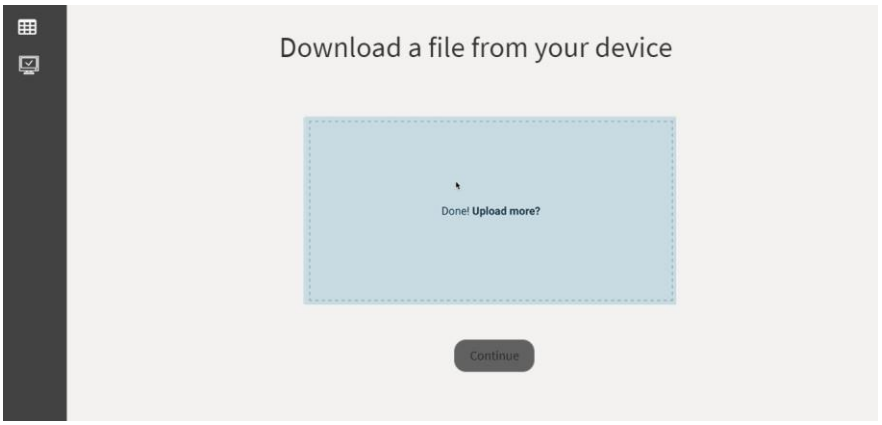


Рисунок 4.31- Дані успішно завантажені

Після того як користувач завантажив всі файли,що він хочу класифікувати, для того щоб їх переглянути слід натиснути на кнопку continue. При її натисненні відкриється вікно з вмістом завантажених файлів (рисунок 4.32). Файли змінювати не можна. Для їх класифікації слід натиснутити кнопку auto classification.Про початок класифікації буде свідчити значок завантаження, що з’явиться на єкрані (рисунок 2.33).

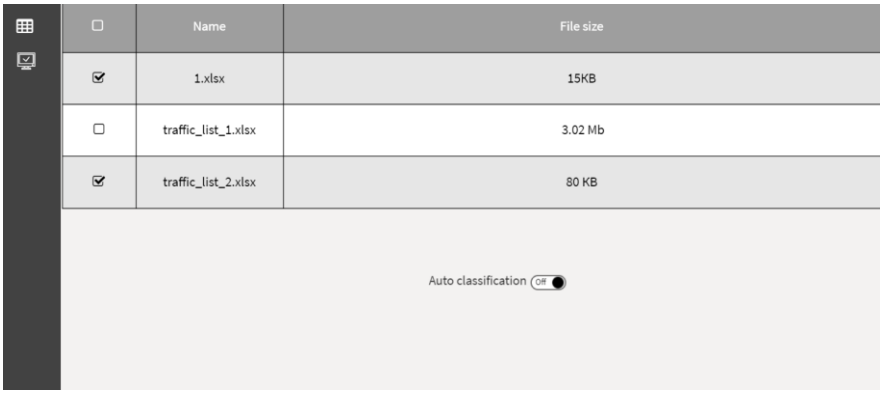


Рисунок 4.32- Вікно перегляду завантажених даних

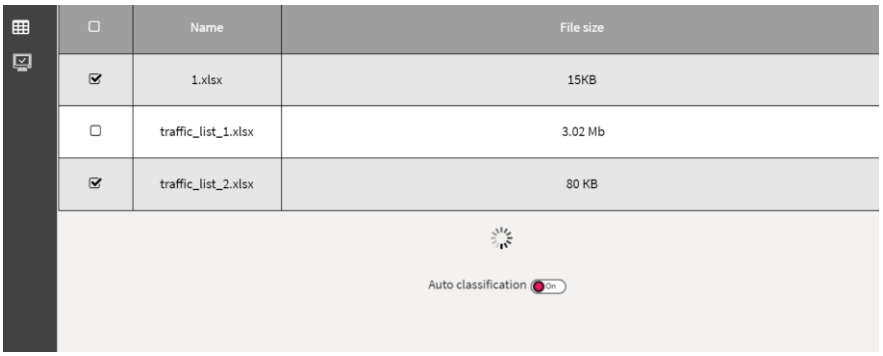


Рисунок 4.33- Процес класифікації

Далі все аналогічно, як і в вікні з введенням даних (рисунок 4.34). Після завершення класифікації, вже класифіковані на класи об’єкти можна завантажити. Після натиснення клавіші new відбудеться повернення до вікна з завантаженням даних для класифікації.

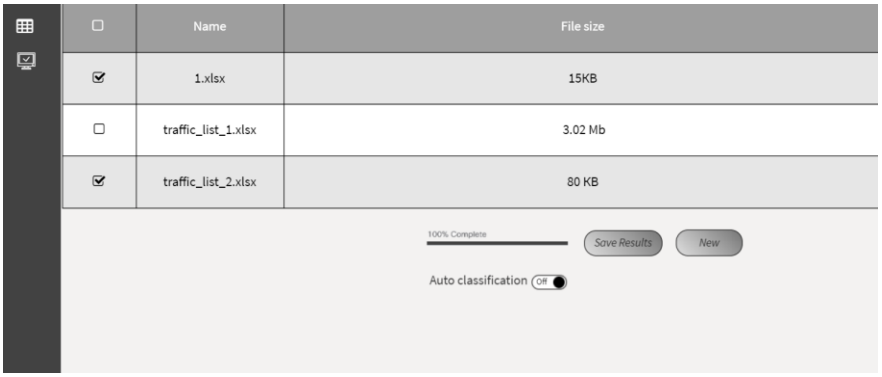


Рисунок 4.34 - Завершення класифікації

# ВИСНОВКИ

Загалом, машинне навчання сприяє пришвидшенню роботи з даними, оскільки технологія допомагає аналізувати великі фрагменти даних, полегшуючи завдання науковцям в автоматизованому процесі і набуває великої популярності та визнання.

Порівняно алгоритми машинного навчання k-найближчих сусідів (KNN), логістичної регресії (LR), випадкових дерев (Random Forest), дерева рішень (Decision Tree) та градієнтного прискорення (Gradient Boosting) для задачі класифікації мережевих втручань.

Можна зробити такі висновки: найкращим та найшвидшим показав себе алгоритм дерева рішень (Decision Tree), він має найменший час узагальнення даних; алгоритм k-найближчих сусідів (KNN) та алгоритм випадкових дерев (Random Forest) другі за швидкістю. А от найповільнішим виявився алгоритм методу опорних векторів(Support Vector Machine), він має великий час узагальнення та не найкращі показники точності прогнозу. Найгірші результати метрик точності і відповідно до метрики AUC має алгоритм логістичної регресії (LR),але варто зазначити,що він доволі швидкий,та все ж не найшвидший.

Розроблено веб-додаток, який дозволяє: класифікувати введені користувачем значення мережевого трафіку і експортувати їх у Microsoft Excel, а потім завантажити вже класифіковані дані у файли того ж формату.

Особлива увага при розробці цього веб-додатку приділено зручності використання та якісному показу даних.

Використання розробленого веб-додатку дозволяє виявити наявність мережевих втручань, шляхом виявлення аномалій в мережевому трафіку.

Використовуючи машинне навчання для класифікації мережевих втручань, можна отримати доволі точні та швидкі результати виявлення аномалій в мережевому трафіку, що підтверджено показаннями математичних метрик оцінки моделей. Таким чином, результати роботи моделей,а саме їх точність в класифікації мережевого трафіку, показують, що пропонований підхід досягає перспективної продуктивності при виявленні мережевих атак на основі інформативних ознак.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Сапожников А. А. Обнаружение аномальной сетевой активности // Доклады Томского государственного университета систем управления и радиоэлектроники, 2009, № 1, с. 79–80.
2. Платонов, В. В. Програмно-апаратні засоби захисту інформації. - М .: Видавничий центр «Академія», 2013. - 336 с.
3. А. А. Браницький, І. В. Котенко, Аналіз і класифікація методів виявлення мережевих атак, Тр. СПІІРАН, 2016, випуск 45, 207-244
4. Гамаюнов Д.Ю. Виявлення комп'ютерних атак на основі аналізу поведінки об'єктів: автореф. дис. ... канд. фіз.-мат. наук: 05.13.11. - М., 2007. - 89 с
5. Baddar S.A.-H., Merlo A., Migliardi M. Anomaly Detection in Computer Networks: A State-of-the-Art Review // Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications. 2014. vol. 5. no. 4. pp. 29–64.
6. K. Alrawashdeh and C. Purdy, ``Toward an online anomaly intrusion detection system based on deep learning," in Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA), Anaheim, CA, USA, Feb. 2017, pp. 195-200.
7. J.P. Anderson, Computer Security Threat Monitoring and Surveillance// James P. Anderson Co., Fort Washington, PA, April. 1980.
8. Воронцов К.В. Машинное обучение: курс лекций / Воронцов К.В. URL:http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0

%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%28%D0%BA%D1%83%D1%80%D1%81_%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D0%B9%2C_%D0%9A.%D0%92.%D0%92%D0%BE%D1%80%D0%BE%D0%BD%D1%86%D0%BE%D0%B2%29

9. Воронцов К.В. Лекции по логическим алгоритмам классификации / Воронцов К.В. URL: http://www.ccas.ru/voron/download/LogicAlgs.pdf
10. Роджерс Х. Д. Теория рекурсивных функций и эффективная вычислимость / Хартли Джером Роджерс. – М: Мир, 1972. – 624 с.
11. Bousquet O., Elisseeff A. Stability and Generalization / Olivier Bousquet , André Elisseeff // Journal of Machine Learning Research. – 2002. – 2. – P. 499-526.
12. Tavallaee M., Bagheri E., Lu W., Ghorbani A. A detailed analysis of the KDD CUP 99 Data Set, Proceedings of the second IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 53–58.
13. NSL-KDD dataset [Електронний ресурс]. URL: http://wwvv.unb.ca/cic/research/datasets/nsl.html
14. Мак-Клар, Стюарт, Скембрей, Джоел, Курц, Джордж. Секреты хакеров. Безопасность сетей - готовые решения, 2-е изд.: Пер. с англ. - М.: Издательский дом «Вильямс», 2001.
15. Wei L, Luan S, Nagai L, Su R, Zou Q. Exploring sequence-based features for the improved prediction of DNA N4-methylcytosine sites in multiple species. Bioinformatics. 2018;35(8):1326-1333.

16. Virtanen P, Gommers R, Oliphant T, Haberland M, Reddy T, Cournapeau D et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods. 2020;17(3):261-272.
17. McKinney W. Pandas: a foundational python library for data analysis and statistics. Python for High Performance and Scientific Computing [Internet]. 2020. Available from: https://www.researchgate.net/publication/265194455_pandas_a_Foundational_Python_Library_for_Data_Analysis_and_Statistics
18. Cario C, Witte J. Orchid: a novel management, annotation and machine learning framework for analyzing cancer mutations. Bioinformatics. 2017;34(6):936-942.
19. Hunter J. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering. 2007;9(3):90-95.
20. The Importance of Machine Learning for Data Scientists | Simplilearn [Internet]. Simplilearn.com. 2020 [cited 13 May 2020]. Available from: https://www.simplilearn.com/importance-of-machine-learning-for-data-scientists-article
21. Durga Prasad Mohapatra, Srikanta Patnaik: Intelligent computing, networking, and informatics. 2014; 821-836.
22. Masleakov N., Boussedjra M., Mouzna J., Labiod H.: C-DRIVE: clustering based on direction in vehicular environment.In: 4th IFIP International Conferenceof New Technologies Mobility and Security( NTMS). 2011; 1-5.
23. Dan Sanderson: Programming Google App Engine with Python. O'Reilly, 2015. 172-178.

24. Park J.H., Stojmenovic I., Jeong H.Y., Yi G.: Computer science and it's Applications. 2015; 293-298.
25. Ivan Idris: Numpy Cookbook. 2015; 75-77.